© 2017 Long Nguyen Thang Le

RESOURCE MANAGEMENT IN SENSING SERVICES WITH AUDIO APPLICATIONS

BY

LONG NGUYEN THANG LE

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Doctoral Committee:

Professor Klara Nahrstedt, Chair
Professor Douglas L. Jones
Professor Rayadurgam Srikant
Professor Venugopal V. Veeravalli

ABSTRACT

Middleware abstractions, or services, that can bridge the gap between the increasingly pervasive sensors and the sophisticated inference applications exist, but they lack the necessary resource-awareness to support high data-rate sensing modalities such as audio/video. This work therefore investigates the resource management problem in sensing services, with application in audio sensing. First, a modular, data-centric architecture is proposed as the framework within which optimal resource management is studied. Next, the guided-processing principle is proposed to achieve optimized trade-off between resource (energy) and (inference) performance. On cascade-based systems, empirical results show that the proposed approach significantly improves the detection performance (up to 1.7× and 4× reduction in false-alarm and miss rate, respectively) for the same energy consumption, when compared to the duty-cycling approach. Furthermore, the guided-processing approach is also generalizable to graph-based systems. Resource-efficiency in the multiple-application setting is achieved through the feature-sharing principle. Once applied, the method results in a system that can achieve 9× resource saving and 1.43× improvement in detection performance in an example application.

Based on the encouraging results above, a prototype audio sensing service is built for demonstration. An interference-robust audio classification technique with limited training data would prove valuable within the service, so a novel algorithm with the desired properties is proposed. The technique combines AI-gram time-frequency representation and multidimensional dynamic time warping, and it outperforms the state-of-the-art using the prominent-region-based approach across a wide range of (synthetic, both stationary and transient) interference types and signal-to-interference ratios, and also on field recordings (with areas under the receiver operating characteristic and precision-recall curves being 91% and 87%, respectively).

To my parents, for their love and support.

ACKNOWLEDGMENTS

This work was supported in part by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP), a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

I am deeply grateful for my committee members, Prof. Klara Nahrstedt, Prof. Rayadurgam Srikant, and Prof. Venugopal V. Veeravalli. They are the smartest people in the world in their respective fields, and as such I'm honored to have them on my committee, to receive, critique, and give insightful suggestions that had profound impacts on my work. I would like to also express special gratitude to Prof. Klara Nahrstedt for her voluntary service as the Chair of my committee, in place of my advisor who could not be physically present during the time of the final examination.

I am eternally grateful to my advisor, Prof. Douglas L. Jones. He is the smartest and wisest mentor that I have ever had in my life. He does not tell me exactly what to do, yet he is always there to give me important advice whenever I need it. He let me grow less-wrong naturally, with a few bruises and burns (figuratively), because he knows that is just the right way to do it. He made sacrifices to give me the best possible opportunities to develop, and waited patiently for me to capitalize on them. One cannot help but feel so blessed to have been received by him.

My friends and lab mates are now my brothers, Dr. David Jun, Dr. Erik Johnson, Dave Cohen, Alex Asilador, Dr. Cagdas Tuna, Jonathan Ligo, and Dr. Rama Ratnam. They helped me in the hardest of times, and we shared big laughs in the best of times. I am blissful to have them as parts of my graduate school experience.

My parents have never stopped supporting me since the day they brought me into this world. Not a single day. I was, am, and will be because of my parents. Yet, I cannot wait for the day that I will be the one to endure

for them, to tell them that it is OK to stop worrying about me and just go enjoying a wonderful, uncaring life, since they have done so much already. I doubt that they would ever agree, but I will insist regardless.

Lastly and most importantly, the lady of my life, Uyen Bui. Ever since we were dating, and eventually got married, she has taught me so much about every aspect of life. She helps me grow beyond my natural zone, to be more thoughtful and caring to everyone around. I always look up to her as the beacon of sympathy and righteousness, whom I vowed to dearly protect in the happiest day of our lives, and ever since.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Sensing and inference systems

Advances in microelectromechanical systems (MEMS) technologies in recent years have enabled sensing devices to be pervasive, allowing data to be collected in new application domains that were previously unavailable. In the mean time, data analysis, inference, and machine-learning techniques have also seen significant development, and they provide necessary tools (e.g. Microsoft cognitive services or Google machine-learning services) to turn collected data into useful information and actions. For example, body-area sensors can collect vital signals such as breathing and heart rates to enable health-care monitoring; an urban-scale sensing network can measure NO2 and CO levels to infer a city's air quality; duplicated/fraudulent vehicle license plates can be detected in real-time with a camera network; a city's noise pollution can be mapped using sound pressure sensors; distributed acoustic sensors in nature parks can help with wildlife and ecosystem monitoring, etc.

Despite major advances in sensing and inference (detection, classification, estimation) technologies individually, the *systematic integration* of the two is still relatively underdeveloped. This observation is articulated in the white paper by the TerraSwarm Research Center (TSRC) [1], and opens up many promising research directions, one of which is undertaken in this thesis. In particular, this thesis investigates resource management problems in a *sensing service*, which is herein defined as a middleware system that takes sensor data as input, and provides inference results (actionable information) as a service to end-applications. It is worth noting that the proposed sensing service is different from normal (joint) sensing and inference systems in that it aims to support multiple applications, i.e. a *shared platform*, as opposed to being tied to a particular one.

1.2 Resource management in sensing services

Solving the resource management problem is believed to be an important step in the realization of the proposed sensing service. Given the number of sensors and the rate at which they generate data, there are simply not enough resources (bandwidth, storage, computing, and energy) to process all data using traditional approaches, especially as Moore's law comes to an end [2]. For instance, traditional approaches include sensors that 1) carry out the inference locally or 2) transmit all data to the cloud for remote inference. However, both of these extremes are unlikely to make the best resource-performance trade-off (generally, the right solution is somewhere in between). Furthermore, both assume that all data must flow through all layers of the system, i.e. one cannot adapt its resource allocation based on the (extracted) information content in the data. This assumption can become a major source of inefficiency in system resource utilization. For example, consider acoustic sensing, where a majority of an audio stream is background noise with little information. Thus, streaming all audio data to the cloud is inherently wasteful. Instead, a better approach would be to have sensors detect acoustic events and only forward these to the cloud for subsequent (event) classification. This observation is later formalized as the *guided-processing* principle.

The merit of the proposed sensing service also depends on its ability to support *multiple applications*. Indeed, in the Internet of Things (IoT), as a system scales up to thousands of sensors across the Internet, it has to support more applications to achieve the *economies of scale*. This general-purpose philosophy is the main factor that sets the proposed sensing service in the IoT apart from its predecessor, i.e. the small-scale, stovepiped/single-purpose wireless sensor network (WSN) [1]. Consequently, efficient resource management within the new multiple-application setting requires novel designs, and it is proposed that sensing services implement the *feature-sharing* principle, in which sharing of features (and thus resources for feature extraction) for related inference tasks is encouraged between applications. The benefits of the proposed principle are later shown to significantly improve the efficiency of the system resource utilization.

To implement the proposed design principles, an *enabling architecture* is needed, in which the sensing-inference process is divided into multiple mod-

ules. Guided-processing can be implemented by having inference results of a module guide the execution of subsequent modules. This principle is not limited to modules arranged in cascade, but also generalizes for trees and (directed acyclic) graphs (DAG) of modules. Thus system resources can be dynamically allocated where they are likely to generate the most value. For multiple applications, related sensing-inference modules (and thus their features) can be shared to conserve system resources, effectively implementing feature-sharing.

While the proposed modular architecture is necessary to achieve optimal resource-performance trade-off, it is not sufficient. Only by carefully optimizing control policies of all modules in a system can synergy be achieved, resulting in a system that is better than the sum of its parts. A contribution of this work is in the development of a principled theory to optimize the proposed system under the stationary data assumption, with approximation results for non-stationary data. The theory is derived to formalize both guided-processing and feature-sharing mathematically, and the basic result is that the optimal decision at each module is obtained by thresholding its posterior probability. Unlike many decision problems, the thresholds in this one play a critical role in the resource/performance trade-off and algorithms for finding their optimal values are also given. The optimization of a sensing service, and the various advantages it has over an unoptimized one, remains unexplored, hence warranting the research direction pursued in this thesis.

1.3 An audio sensing service

The abstract framework developed earlier is applied to create a prototype of an audio sensing service. While the existing inference/machine-learning techniques (deep neural network, support vector machine, hidden Markov model, etc.) can be used within the sensing service, their predication on the availability of a large *labeled* training dataset for successful performance limits their usage in this multiple-application setting. (Except for speech-recognition, the majority of applications, e.g. wildlife recognition, machinery classification, etc., simply cannot afford to have hundreds hours of labeled datasets.) Hence, a novel template-based classification algorithm is introduced to address this issue. The technique is a combination of AI-gram

time-frequency representation and multidimensional dynamic time-warping. Compared to the existing toolbox, the proposed technique is more suitable for sensing services, as it not only requires very few training data, but is also more robust to interferences.

This thesis is organized as follows. Chapter 2 presents the modular architecture of the proposed sensing service in detail, followed by a broad review on the optimization of (distributed) sensing and inference systems in Chapter 3. Then the guided-processing principle for single-application systems is introduced in Chapter 4, and its generalization to the DAG topology is given in Chapter 5. Next, the multiple-application setting is considered and the feature-sharing principle is introduced in Chapter 6. Finally, the robust, template-based audio classification technique that is designed specifically for sensing services is presented in Chapter 7. Final remarks and conclusions are given in Chapter 8.

CHAPTER 2

SYSTEM ARCHITECTURE

2.1 Related works

Existing sensing and inference systems in the literature can be grouped into the following four architectural types. The first type includes systems in which sensors do all the processing and only send results back to a server for visualization. In contrast, sensors in the second type only collect measurements and send all data back to the cloud for processing. Since transmitting all data can be costly for bandwidth and power, it is more efficient for a sensor to extract features, i.e. compressive transform of the data, and send them instead. This is the approach of the third type. Note that the systems mentioned so far are relatively straightforward, with only a data path and no control path. The fourth system type introduces the control logic, in which sensors can use the result of an inference unit, e.g. an anomalous event detector, to make data-driven decisions on when to transmit data to the cloud.

It is natural to question the trade-off of these systems relative to each other. While a *quantitative* comparison between them heavily depends on the underlying implementation and the particular application, a *qualitative* comparison based on certain assumptions is still of interest. Figure 2.1 summarizes a qualitative comparison between different systems based on the processing and transmission requirements on sensors, as well as the storage and processing requirements on the cloud. The white space in Fig. 2.1 reveals the opportunity for a new architecture to improve over existing ones. More detailed discussions about each architecture are given in the following sections.

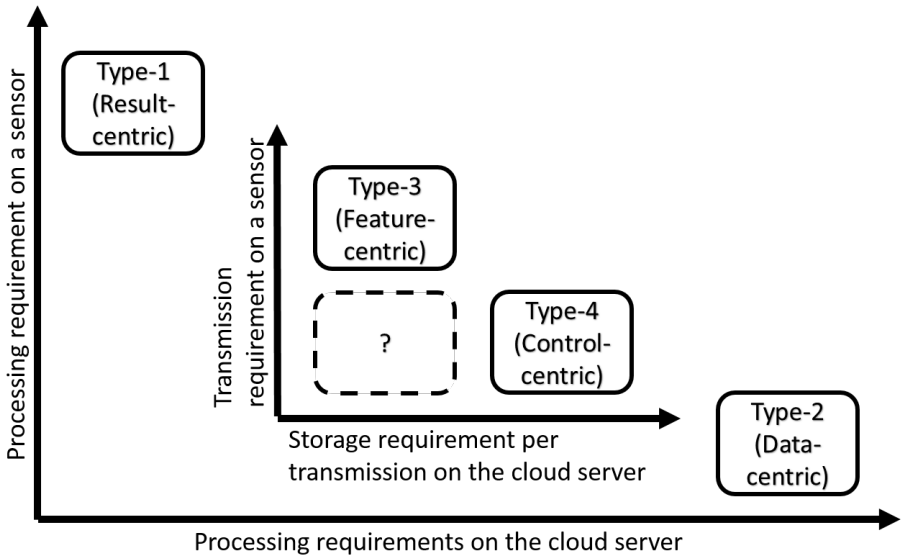


Figure 2.1: Comparison of different architectures based on resource requirements. The white space reveals the opportunity for a new architecture to improve over existing ones. Section 2.2 presents the proposed architecture to achieve this.

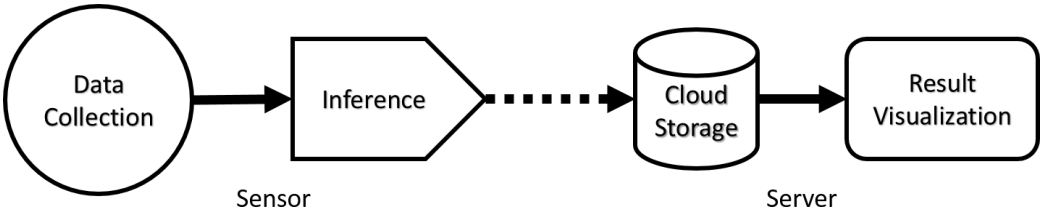


Figure 2.2: Type-1 system where sensors collect and process data; the server is only used for result visualization.

2.1.1 Type-1 (Result-centric)

This architecture (see Fig. 2.2) is the least resource-demanding for the cloud, since sensors do all the processing and only send results back to the cloud for visualization. The following work took this approach.

Lu et al. created SoundSense [3], a large-scale sound-sensing system using mobile phones. In [3], it was recognized that building an automatic sound classifier for all possible ambient sounds on resource-limited phones is infeasible, and they proposed to personalize the system to individual users. SoundSense achieves this by classifying sound hierarchically, with a coarse, categorical classification followed by a finer, intra-categorical one. The coarse

category is the same for every user, including music, speech, and ambient sound, and the classification is carried out using a decision tree on a frame basis with features such as zero-crossing rate, spectral flux, etc. The final decision is smoothed using a first-order Markov model. If speech or music is detected, it can be further analyzed using standard techniques developed for those audio types. For the generic ambient sounds, an unsupervised clustering based on a Gaussian mixture model is employed to group related sound-classes together. [3] also shows that the *significance* of each ambient sound class, defined based on the frequency of encounter and duration, is user-specific, and users are prompted to label the clustered sound classes manually. It is worth noting that all the processing is done on the mobile phone and no back-end is used. The resource-constraint problem of the mobile platform is handled in the preprocessing step with an audio-frame admission controller, with only frames of high energy or spectral entropy being admitted for further processing.

The Jigsaw engine is another mobile-phone platform for continuous sensing with accelerometer, microphone and GPS sensors [4]. In Jigsaw, processing pipelines for the accelerometer and microphone data are built to convert raw time-series data into recognized activity and audio, respectively. The pipelines share similar steps including the framing of data, frame admission control, feature extraction and classification, and final output smoothing. The location pipeline for the GPS sensor is handled differently. Recognized activities from the accelerometer pipeline are fed into the GPS pipeline to make an informed decision about the GPS duty cycle that optimally trades between location accuracy and energy consumption of the phone. The decision-making process is viewed as a completely observable Markov decision process, with the states being the recognized activities, and actions being the GPS duty cycle. It is well known that the optimal decision can be found using the dynamic-programming algorithm/Bellman equation [5]. The entire Jigsaw engine is run as a background service on the phone, allowing continuous sensing and provision of inference results (activities, audio classes, location) to other applications and services in the system.

Rana et al. created Ear-Phone, an urban-noise-mapping system based on participatory sensing from mobile phone owners [6]. The system consists of mobile phones, a central server, and end users. The mobile phone runs an Ear-Phone signal-processing module that computes the A-weighted equiv-

alent sound level $\mathrm{LA}_{\mathrm{eq},T}$, where T is the measurement interval, using the on-board microphone. It also uses the built-in GPS receiver and system clock to obtain and attach a time and space stamp, in latitude and longitude, to the computed noise level before being stored in the phone memory. Every two minutes, stored data are offloaded to the central server over the Internet. Since reports from mobile phones are sporadic and stochastic in general, the signal reconstruction module running at the central server is used to reconstruct the sound field given the incomplete, limited samples of the urban sound fields reported across space and time by mobile owners. This is achieved by solving a compressed sensing problem assuming the actual noise profile is sparse in the DCT domain and applying the standard l_1 minimization. The output of the reconstruction is comparable to a commercial off-the-shelf (COTS) sound-level meter if there is a minimum number of phones participating, which was found empirically to be four for the noise profile in their experiment. Finally, the reconstruction is presented to the end user via a web interface.

Dutta et al. [7] designed a large-scale (10000 nodes) WSN that detects the rare, random, and ephemeral presence of civilians, soldiers, and vehicles using multi-modal, i.e. passive infrared, magnetic, and acoustic, sensor nodes. Nodes in the system achieve long battery lifetime (1000 hours of continuous operation) by using both duty-cycling and wake-up mechanisms. The wake-up mechanism works by having the PIR sensor trigger the microphone, the magnetic sensor, and the fusion algorithm. Based on the output of the acoustic and the magnetic sensors, the fusion algorithm classifies a target based on simple logic; i.e., it is assumed that the presence of civilians does not trigger acoustic and magnetic sensors, while soldiers can trigger acoustic but not magnetic sensors, and vehicles can trigger both. The result is then transmitted back to a server.

IrisNet [8] is among the earliest forms of a sensing service for information-rich sensor data with high bit-rate, e.g. video streams. Specifically, IrisNet's design principle highlights the importance of a shared sensing infrastructure that abstracts away the complexity of distributed sensor networks from application developers, offering data as a "single queriable unit" [8]. IrisNet's approach is to push all application-specific processing to the sensor nodes by introducing the concept of application senselets. Senselets distill the raw data stream into semantic information that is then stored in distributed databases

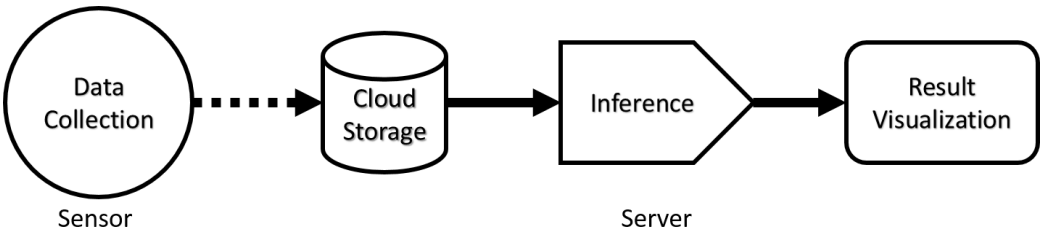


Figure 2.3: Type-2 architecture where sensors only collect data and all processing are done on the server.

where it is later used by application front-ends without further processing. To address the limited resource problem on sensor nodes, IrisNet offers a common execution environment for senselets where common processing routines and their partial results are shared to avoid duplicated work.

More recently, the Array of Things (AoT) project, which is a large scale deployment of environmental, air quality, and light/infrared sensors, was introduced in Chicago with the goal of capturing the "fitness" of the city [9]. It is envisioned that by making the data open, third parties can contribute to improve the livability of the urban landscape. For example, urban planners can enact data-driven policies to improve traffic efficiency, and innovative developers can create new apps that offer their users real-time awareness of air contamination, urban heat islands, noise and traffic congestion. However, in terms of the audio sensing modality, the system architecture is relatively simple, with only sound intensity being computed at edge nodes, to be subsequently transmitted to a public database for visualization. The same goes for the video sensing modality, in which all processing is carried out on sensor nodes (with the exception of some images, i.e. 1%, being uploaded for calibration).

2.1.2 Type-2 (Data-centric)

Sensor nodes in this architecture (see Fig. 2.3) only have to collect and forward data, hence very limited processing is required on sensors. However, processing data streams from all sensors on the cloud is resource-demanding. Therefore, this architecture is often limited to low-data-rate applications only.

The Harvard forest is home to a collection of multimodal sensors (atmo-

spheric, temperature, humidity, pressure, pheno-camera[1], acoustic, etc.) that aims at monitoring the entire forest ecosystem [10]. In particular, the three acoustic sensors are Raspberry Pi-powered, Linux machines that records sounds at 24 kHz. The data are then streamed wirelessly (via Wi-Fi) over the Internet to remote hard drives for further development of algorithms to automatically identify forest (i.e. bird and insect) species. It is worth noting that this audio streaming is only meant as a pilot study, not as a long term deployment, as clarified by the researcher on the receiving end of the data in the following statement "Nobody is going to sit and listen to terabytes of sound data" [10].

Air quality egg (AQE) is an open-source hardware platform for environmental sensing [11]. The system includes three components. The first component is an AQE sensor node, which is comprised of two units: a base unit that has a wired connection to the Internet, and a secondary unit that collects NO_2 and CO levels which get reported to the base unit every few minutes using a custom wireless protocol. The second component is the distributed database Xively [12] in which all data forwarded from the base unit are stored, and the last component is the AQE website where the data are publicly accessible. There are variants of AQE outside the US, such as Alima (France) and iKair (China).

For indoor sensing, Jiang et al. [13] designed and implemented ACme, a wireless sensor and actuator network for monitoring AC energy use and controlling AC devices. The system consists of three layers: ACme nodes whose interface offers the ability to read the energy usage and provides on-off control of AC devices, a network fabric that extends the node interface to arbitrary IP end points, and a web application that utilizes the network fabric to visualize the energy data. ACme nodes are configured to report energy readings back to the cloud, along with average, minimum, and maximum instantaneous power every minute for visualization.

2.1.3 Type-3 (Feature-centric)

In this architecture (see Fig. 2.4), sensors perform feature extraction on incoming data streams. Since features are compressive transforms of the raw

[1]For the study of natural cycles.

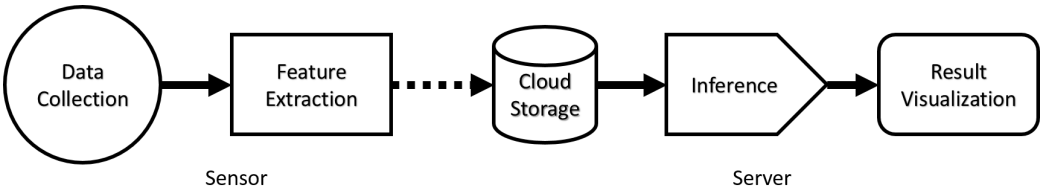


Figure 2.4: Type-3 architecture where sensors extract and transmit features to the cloud for inference.

data, it helps lower the storage requirement per transmission on the cloud server. Note that the transmission (power and bandwidth) requirement is still formidable because the transmit rate is high. Furthermore, feature extraction also helps offloading some of the processing requirements on the cloud server onto sensors. The following work uses this architecture.

With the goal of capturing the sound landscape of New York City, the SONYC project introduced a blueprint for an urban-scale audio sensor network [14]. By the time of this writing, the design and implementation of the SONYC sensor node, which includes a Linux-based miniPC (1.6 GHz quad core CPU, 2 GB of RAM, 8 GB flash drive, USB I/O, and WiFi connectivity) and an omni-directional analog MEMS microphone (Knowles SPU0410LR5H-QB) connected to the minPC through a USB CODEC, has been completed. A power supply line is required to operate the node. At the moment, the sensor node continuously samples 16 bit audio data at 44.1 kHz in 1 minute segments. These segments are compressed using the lossless FLAC audio encoder, which can be thought of as a trivial form of feature extraction, before being sent (encrypted) back to the server once local (backup) storage is full (approximately two days). However, future plans for SONYC seem to be heading toward Type-1 architecture, where it was proposed that a deep-learning-based audio classification (whose model is likely to be compressed to fit the miniPC) is executed entirely on-node, and only results are sent back.

2.1.4 Type-4 (Control-centric)

The architectures (see Fig. 2.5) discussed so far are straightforward, with only the data path and no control path. Having a control logic in a system enables dynamic, data-driven adjustment of its resource allocation to improve

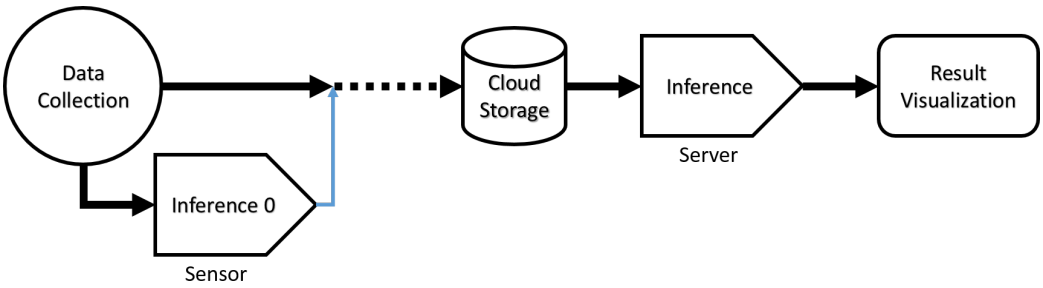


Figure 2.5: Type-4 architecture where a sensor uses the result of its limited inference unit, i.e. Inference 0, to detect information-rich events from a data stream and control (blue thin line) the data transmission to a server database. The complete inference is then carried out on the server.

efficiency. Hence, a control path is introduced in Type-4 architectures as shown in Fig. 2.5. In particular, a sensor performs a limited inference task, i.e. Inference 0 from Fig. 2.5, to identify and transmit only information-rich data from its data stream. The sensor's inference task requires additional processing, but its result helps reduce both the transmission rate and the processing requirement on the cloud. However, the selective transmission of raw data still demands significant storage per transmission. Systems that use this architecture are described below.

Many large-scale speech-recognition systems such as Apple's Siri [15], Microsoft's Cortana [16], and Amazon Echo [17], have voice-activity detectors on their sensing devices. The detector identifies speech instances from an audio data stream and only transmits these to the cloud for further processing, i.e. recognition.

Recently, in video sensing, Jain et al. created Overlay, a real-time, scalable augmented-reality system on mobile devices [18]. Overlay offers the ability to retrieve the relevance tag for the current image (object) in a video stream, with high accuracy in real-time. The system has two modes. In the annotated mode, mobile phones are used to crowdsource and build an annotated image database on the remote server. Well-known computer-vision techniques are used to reconstruct the 3D camera pose from the retrieved 2D images in order to cluster and prune down annotated images from similar poses. In the real-time retrieval mode, image frames captured by the camera are first screened to remove near-duplicate and blurred frames. This is achieved with the assistance of the phone's accelerometer and gyroscope along with a Canny edge detector. Frames that make it through are sent to a remote server

where they are matched against a prioritized set of candidate annotated images using the SURF feature [19]. It is critical that the candidate images from the large image database are prioritized for the system to perform in real-time. The prioritization uses a simple, yet effective, predictive model of users' trajectories to identify objects that a user is most likely to encounter next, given her current (relative) location. The trajectory is based only on the orientation, which is sensed using the gyroscope, and the time elapsed between objects encountered by the user. Note that GPS-based localization is not a feasible option due to its high power requirements and low accuracy in indoor environments. The overall system results in good precision and recall rate, while having low latency.

2.1.5 Service-oriented architecture in the Internet of Things (IoT)

The fast growth of wireless and embedded technologies in recent years has unleashed the new Internet of Things (IoT) paradigm, in which sensing, computing, and networking capabilities are envisioned to be pervasive. Unlike previous small-scale sensing-inference systems that operate on ad-hoc networks and often focus on a single application, systems in the IoT are distributed across the Internet and are expected to be a platform that can support multiple applications, to achieve the *economies of scale*. Therefore, designs for the IoT should be modular, allowing modules to combine and form various applications with different objectives and constraints. This observation motivates the *service-oriented architecture* for the IoT [20], where an ecosystem of services lays the foundation for IoT applications to be built on top. A similar view is shared by the TerraSwarm Research Center, whose aim is to create software components that serve as building blocks for IoT application developers [1].

In a service-oriented architecture, sensing is also a service [21, 22] where applications can indirectly obtain data from the underlying sensor networks. In [21], a sensing service is a cloud service that collects and stores sensed data from participating smartphones. Multiple applications can then request the cloud service for sensor data. Perera et al. in [22] propose a more detailed model for the SaaS architecture with a four-layer architecture. At the bottom

of the stack is the sensors layer. The second layer is the sensor publishers layer, where registration and discovery of new sensors are also handled. Data generated from sensors are also managed in this layer, e.g. kept private or published to public domains according to the sensor-owner's policy. The third layer contains extended-service-providers, which serve as the interface between data consumers and public sensor data. An example of a service provider is the sensor search service in [23], where relevant sensors are selected for data out of a large number of sensors with similar capabilities. Finally, the sensor data consumers layer is where data-driven applications reside.

2.1.6 The psychological theory of visual guided search

Many biological systems have been optimized through the driving force of natural selection for many thousands of years. Therefore, it is quite common for modern engineers to take inspiration and insight from these systems and apply them to solve engineering challenges. Relating to the resource optimization problem, it is known from the psychology literature that the human visual system, despite being capable of performing very complex vision-related tasks, is unable to process all the data presented to it. This suggests that the human brain is very resource-efficient in its processing. The celebrated psychophysical model that explains how this non-trivial feat can be achieved is briefly summarized here.

The visual guided search model [24, 25] is a powerful psychophysical model of the way the human visual system works, specifically for search tasks. It is observed that the human visual system can search for targets in a crowded visual world with relative ease, even though "there is not enough room in the skull for all the neural hardware that would be required to perform all visual functions at all locations in the visual field at the same time" [24]. The model claims that this non-trivial feat can be achieved in two stages: a preattentive and an attentive stage, with an important observation that the information in the preattentive stage can be used to guide the deployment of attention in the later stage. At the center of the preattentive stage is a set of coarse, categorical channels. Depending on the way channel responses are used, there are two types of guidances: bottom-up and top-down.

An example of a bottom-up guidance can be seen from Fig. 2.6, which

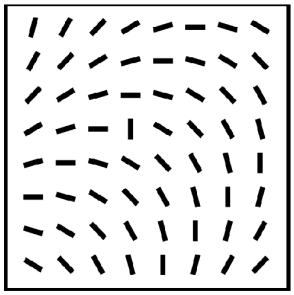


Figure 2.6: An example of bottom-up guidance.

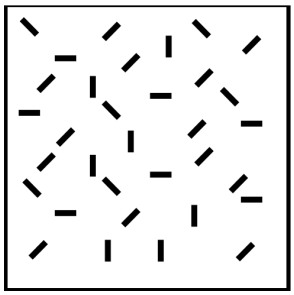


Figure 2.7: An example of top-down guidance.

is adapted from Fig. 8.4 of [25]. Even without any instructions in advance about which item to notice, the vertical line in the middle captures one's visual attention. This is the result of a significant difference in orientation between the item and its neighbors. Note that there are four other vertical lines in the figure, but these are not conspicuous because they do not have strong local contrast. In other words, bottom-up guidance uses the *difference* between channel responses of an item and its neighbors.

The second type of guidance is top-down or task-based, which is based on the *absolute* channel response of an item. For example, consider a set of categorical channels that includes red, yellow and green, meaning they respond most strongly to red, yellow and green items, respectively. Then the task of searching for orange among yellow items can be done efficiently simply by inspecting the response of the red channel. This is because all items respond to the yellow channel, no item responds to the green channel, and only orange items respond to the red channel. The top-down guidance is useful when bottom-up guidance does not exist, i.e. there is no significant local contrast in the target item. This is illustrated in Fig. 2.7, adapted from Fig. 8.5 of [25]. There is no significant local contrast in this figure,

but efficient search can still be done given the task of finding only horizontal items, for instance.

To guide the deployment of attention in the attentive stage, the bottom-up and top-down guidances are combined to create the so-called *activation map*. In guided search [25], the combination is simply a linearly weighted sum,[2] i.e.

$$A_i = w_B B_i + \sum_n w_n C_i(n), \quad \forall i \tag{2.1}$$

where

$$B_i \propto \sum_{j \in \text{neighbor}(i)} \max_n C_i(n) - C_j(n) \tag{2.2}$$

The subscript i denotes the i-th item in the visual field/map. A is the activation strength, B is the bottom-up guidance, and $C(n)$ is channel n's response. The w_B and w_n are the weights associated with the bottom-up guidance and channel n, respectively. The activation map is effectively a ranking to guide attention deployments on the visual field.

Even though the discussion so far is confined to the visual system, similar notions can be adopted for the auditory system, as was done in [26]. In [26], Kim et al. remark that acoustic-based attention can be driven either from the top down by intention or bottom-up by *salient* stimuli.

2.2 System design

Building on the merit of previous works from Section 2.1 and borrowing intuitions from the guided search model, we propose a system architecture for integrating sensing and inference that can support multiple applications simultaneously in a resource-efficient manner. The key idea is the division of the sensing-inference process into smaller modules, each of which uses the result of its inference to guide the execution of subsequent ones. We name this feature *guided processing*. This approach improves resource efficiency by allowing limited resources to be dynamically allocated where they are likely to yield the most value. Furthermore, the modular architecture also encourages different applications to cooperate by sharing the results of common modules,

[2] In GS 4.0, there is an additional noise term that is dynamic, but this is not important for our adoption of the GS model.

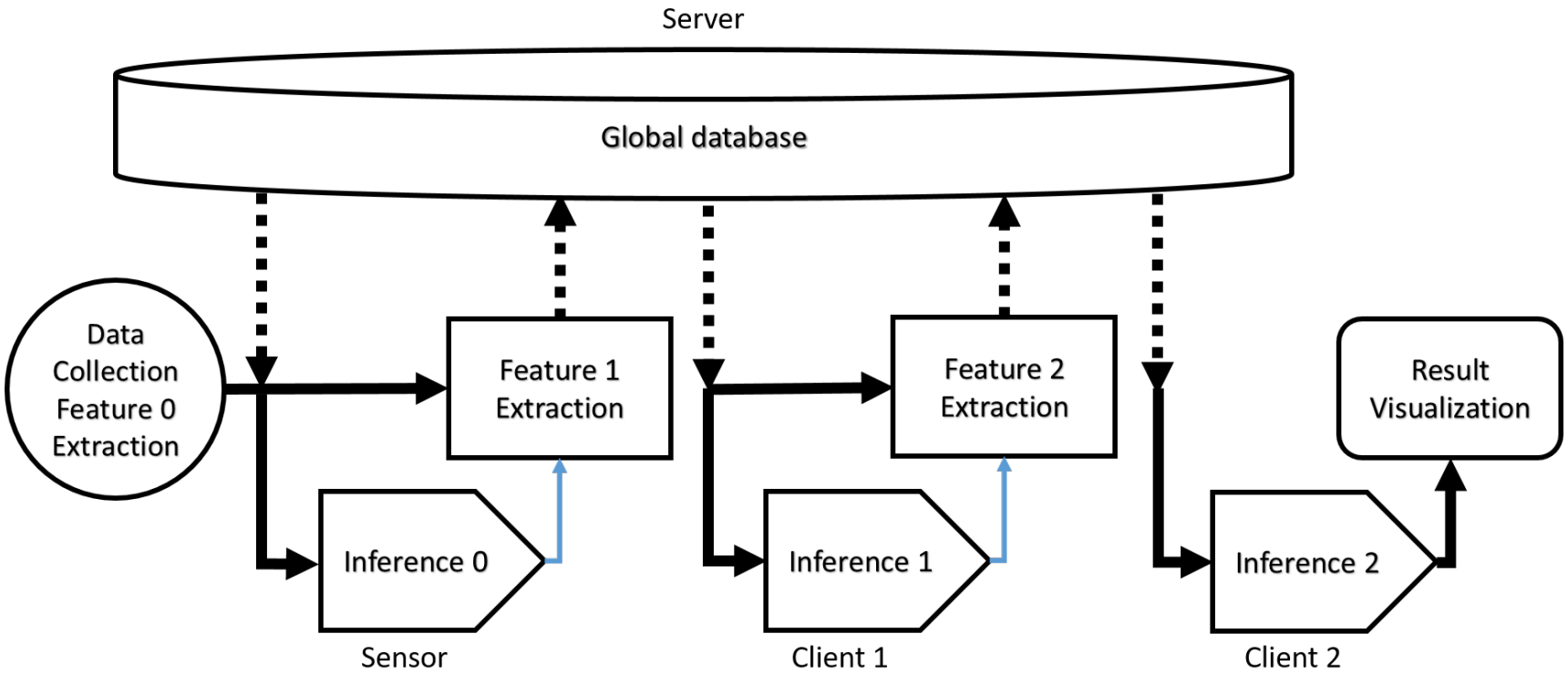


Figure 2.8: In the proposed architecture, the main server hosts the global database. The segmented line represents a remote connection to the database from multiple sensing-inference modules. In each module, the data path is denoted in thick black, and the control path in thin blue. Intermediate inference results are used to control the generation of new features (i.e. Inference 0 controls the feature for Inference 1, which in turn controls the feature for Inference 2), thus guiding the execution of subsequent modules. We name this property guided processing. Another important property is *feature sharing*, in which features produced by modules of an application can be reused by those of others, by accessing the common database. This design is motivated by the observation that a feature might be useful to more than one applications.

which is referred to as *feature sharing*. The collaboration is based on an observation that a feature might be useful not only for one application but also others.

Figure 2.8 shows a realization of the above idea. In the proposed architecture, the main server hosts the global database of the system. In addition to being a storage space, the global database also serves as a communication medium for all modules of the system (see [27] for more details about a global data plane (GDP) log, a joint communication-storage infrastructure). Namely, with proper authorization, modules can access the database to read data output by others, perform inference, and write back newly generated data. Unlike sensor modules, which have exclusive access to a data stream, other modules hosted on general-purposed processors (i.e. Clients 1 and 2 in Fig. 2.8) depend on features output by other modules. Hence, each module in the system can guide the execution of subsequent modules by controlling the generation of new features with the result of its inference. Note that this design is basically a combination of Type-3 and Type-4 architectures to fulfill the white space in the trade-off space of Fig. 2.1; i.e., it achieves both the low transmission requirement on modules and the low storage requirement per transmission on the cloud. Finally, the last module of an application does not generate new features, but simply displays its inference result.

Applications can be implemented by combining modules, and multiple applications can share common modules. For instance, a speech-recognition application can be implemented by combining the following modules: an acoustic sensor, a speech-activity detector, and a speech recognizer. The sensor module performs anomaly detection and writes the detected audio clips to the global database. The speech-activity detector can then access these raw data clips from the database to detect speech instances, for which MFCC features are extracted and written back in the database. Finally, a speech recognizer can access the extracted MFCC features from the database and process them using a deep neural network trained for speech recognition. Next, a speaker-identification application can reuse the outcome of both the acoustic sensor and the speech-activity detector, and only needs a new speaker-identifier module. Finally, other audio applications such as wildlife monitoring can also reuse the acoustic sensor module. In these applications, the sensor module can select compressive features to log instead of raw audio clips, which helps further reduce the communication bandwidth.

Then other modules can access the database and process these features using an application-specific inference engine.

In summary, the modular architecture in Fig. 2.8 has multiple sensing-inference modules interacting with a cloud-based global database to form applications. More detailed discussions of each system's component are given in the following sections.

2.2.1 Cloud server

Our cloud server is a static-IP machine,[3] which is a ML100G-10 next unit of computing (NUC) from Logic Supply that has a 4-core, 1.83 GHz processor, 8GB DRAM, 64 GB SSD in a fanless enclosure. The global database, hosted on this server, is the most important component of the proposed system as it not only provides the data storage capability, but also functions as a communication medium for modules in the system. We implemented the global database using MongoDB [28], which is an open-source software framework for deploying noSQL/document databases with seamless support for horizontal scaling,[4] i.e. simply adding more NUC if its 64 GB storage runs out. In MongoDB, data points, which include both *records*, i.e. key-value pairs,[5] and *blobs*, i.e. binary large objects, are not stored in tables, but encapsulated as individual/independent units (i.e. documents), and can be grouped into collections. Since records can be indexed and searched by the database, while blobs cannot, they are separated into two collections. However, this arrangement does not preclude hybrid data points, which have both the record and the blob components, as they can be implemented by having a field in the record points to the corresponding blob's ID. In fact, all acoustic data in our prototype are hybrid data points.

Since we want the global database to be easily accessible to all modules remotely, HTTP is used as the communication protocol and a custom RESTful application programming interface (API) is implemented to wrap around the database. The API includes three main servlets. The record-servlet handles the read, write, update, and delete of the record of a data point by its unique

[3] `acoustic.ifp.illinois.edu`

[4] Although the support for vertical scaling in MongoDB was often the source of criticism in the past, with the introduction of the document-level locking scheme in version 3.0, a major improvement in MongoDB's vertical scalability has been delivered.

[5] Often written in the JavaScript object notation (JSON)

ID.[6] The update operation allows a data point to expand, i.e. adding fields to a record, with new output from modules that processed the data. Similarly, the blob-servlet handles the read, write, and delete of a data point's blob (blob cannot be updated). Lastly, the search-servlet handles database searches for data points that match certain criteria.

Finally, for security and privacy reasons, the API requires authentication for all database accesses at the data-point level. Namely, access to a data point is only granted if the requester's key matches with the key signed on the data by its writer. Examples API are as follows:

/col?dbname=⟨database name⟩&passwd=⟨key⟩&filename=⟨data point's ID⟩

reads from the named database the record of a data point by its ID, with a given key. Similarly,

/gridfs?dbname=⟨database name⟩&passwd=⟨key⟩&filename=⟨data point's ID⟩

reads from the named database the blob of a data point by its ID, with a given key. The complete API and example codes are available online.[7]

2.2.2 Sensor module

Embedded systems, on which sensor modules are hosted, generally have stringent resource (bandwidth, storage, processing, and energy) constraints. For instance, we experimented with Samsung Galaxy S2 smart phones (dual-core, 1.2 GHz Cortex-A9, 1GB RAM) and the WGM110 embedded chip (48 MHz Cortex-M3, 128 kB RAM + 1 MB Flash) as sensor modules. Hence it is desirable to keep the processing on the sensor module generic and efficient, while leaving application-specific and intensive tasks for subsequent modules. For the remainder of this section, we show how this can be achieved for the acoustic sensor.

[6] In the current prototype, a data point's ID is a hash of its timestamp and the MAC address of the generating sensor.

[7] `https://github.com/longle2718/sas-client`

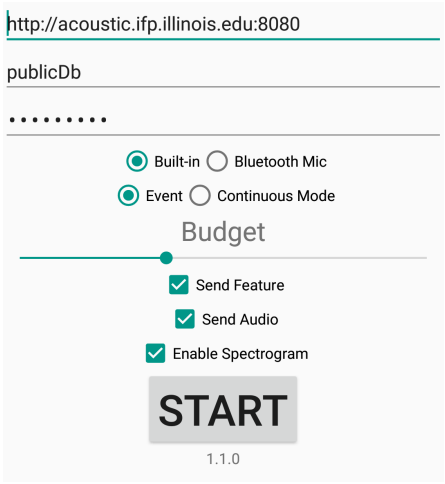
2.2.2.1 Inference task

The input into an acoustic sensor is a high-rate (16 kHz) data stream, which contains both noise-only and (noisy) signal instances. The goal of the sensor's inference task is to detect signal instances, i.e. be a signal event detector, so that feature extraction for subsequent processing can be controlled/guided to focus on information-rich data, thus improving the system efficiency of resource utilization. Note that without any guidance/control, feature extraction must be carried out periodically for all data.

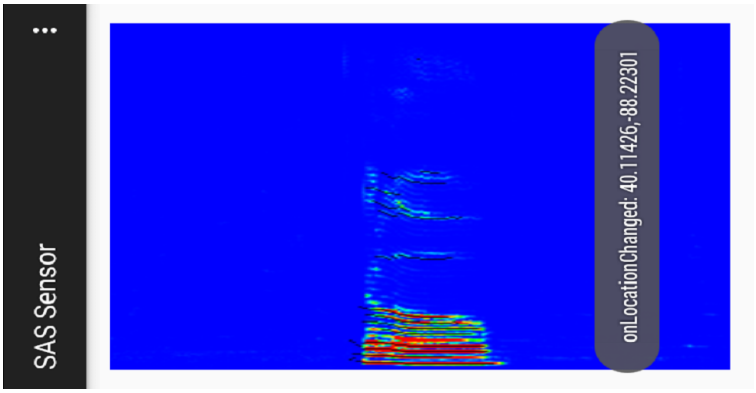
In many applications, acoustic (signal) events often exhibit unique time-frequency (TF) structures that, if known, can streamline the detector design. Fortunately, there exists a vast literature on techniques for estimating the TF structure of an audio signal, by means of tracking its TF ridges, also known as McAulay-Quatieri (MQ) sinusoidal tracks [29, 30], instantaneous frequency (IF) tracks [31], time-varying harmonics [32], etc. However, there are drawbacks in existing techniques. The tracker in [31] can only track the highest power frequency component; the particle-filter-based solution in [32] is computationally intensive; and the classical MQ tracker in [29, 30] uses a suboptimal, greedy approach. Consequently, for the sensor module implementation, a custom technique is proposed that overcomes the above limitations and extracts/estimates prominent (high-energy) TF ridges. Details are given in Section 7.3.1.

2.2.2.2 Feature extraction

In the multiple-application setting, it is important that the feature extraction unit select *general-purpose features*, for which there are multiple choices. An obvious choice for such a feature is simply the raw audio data of detected events, but uncompressed features are demanding in both communication bandwidth and transmit power. In addition, unlike raw data, compressed features have semantics, and can be indexed and searched in a database. Fortunately, there are better options for compressed features. For speech-based applications, the mel-frequency cepstrum coefficient (MFCC) feature can be chosen. For other applications, we propose the TF ridges mentioned earlier as the feature, since they strongly characterize many acoustic events and are readily available. All of these feature options are available in the



(a) Main menu of the sensor app.



(b) Running spectrogram of the input audio stream. The detected TF ridges are overlaid as black lines.

Figure 2.9: Sensor module implemented on Android devices.

prototype implementation of the sensor module.

The sensor module is implemented as an app for Android devices. Figure 2.9a shows the main menu of the app.

- The first three lines contain the address of the main server, the name of a database available on the server, and the sensor's authorization key (to be signed on all data it produces), respectively. Note that there can be multiple databases hosted on the same server, but only modules working on the same database can collaborate.
- The app allows flexible selection of an audio source, either from the built-in microphone, or an external Bluetooth one.
- As alluded to earlier, there are two modes of feature extraction: one

controlled by an event detector, and the other by a periodic (every minute) timer.

- An (energy) "Budget" slider is used to input the relative, i.e. lowest (leftmost) or highest (rightmost), resource available for the app (More details are given in Chapter 4).

- Output options include compressed features and raw audio data. Features consist not only of the audio features (MFCC, TF ridges, and sound pressure levels in N octave-bands), but also metadata such as an event's timestamp, a sensor's location, etc. (see Listing 2.1). Output are first logged to the local storage and then uploaded using the HTTP protocol to the server as soon as network connectivity is available. To ensure reliable data uploading, the sensor waits for the server's acknowledgement of successful reception before clearing its local copies. Otherwise, retransmissions are attempted.

- Finally, while all the processing is done as a background service, the app can optionally display the spectrogram of the incoming audio stream and overlay it with detected TF ridges, shown as black lines in Fig. 2.9b.

Listing 2.1: A sample event's record. Audio features are highlighted in red.

```
GCWA_score: 0.0639
MFCCFeat: Array(59)
TFRidgeFeat: Object
androidID: ''b8a9953125a933af''
avgSNR: 28.93277114284566
batteryLevel: 100
blkSize: 512
device: ''Android''
filename: ''c702e60a-351d-4742-a76b-18b2aafbbe70.wav''
freqSize: 256
fs: 16000
incSize: 256
key: ''publicPwd''
location: Object
```

```
maxDur: 0.912
maxFreq: 4843.75
minFreq: 0
octaveFeat: Array(1)
recordDate: ''2017-04-16T18:55:39.538Z''
serviceAddr: ''acoustic.ifp.illinois.edu''
_id: ''58f3be2d0fffccaf30d45d9d''
```

2.2.3 Client modules

Client modules are modules hosted on general-purpose machines. For instance, we experimented with a NUC (4-core, 1.83 GHz processor, 8 GB RAM), a laptop PC (4-core, 2GHz processor, 8 GB RAM), and a desktop PC (8-core, 3.6 GHz processor, 32 GB RAM) as client modules. In the future, we hope to experiment with the recently debut Amazon Lambda service, which offers a serverless business/computing model, as an economical/efficient platform for hosting client modules. These machines do not have exclusive access to a local data stream and must (remotely) query the server database for features generated by other modules, e.g. the sensor module. The data returned from the query form a dataset on which the module's inference engine can train/test. The particular model and training algorithm for an inference engine can directly benefit from the vast literature on machine learning/statistical signal processing. For instance, an inference engine can be a support vector machine (SVM), a deep neural network (DNN), or a deep belief/Bayesian network (DBN), etc. In addition, the processing module can optionally produce new features for subsequent processing using well-known algorithms from the same literature, such as principal component analysis (PCA), independent component analysis (ICA), or autoencoders, etc.

2.2.4 User interface

Figure 2.10 shows the graphical user interface (GUI) of the proposed system, implemented as a web page in web browsers. The GUI abstracts away the underlying API and allows an end-user to interact with the proposed system at a high level. There are three panels in the GUI: the left one is for user

input, while the right ones are for result visualization. The upper and lower right panels show detected acoustic events/data points in space and time, respectively. Clicking on a data-point automatically invokes the retrieval of its raw audio data (if available) to be played, and opens a dialog box from which the user can either provide a label or download the audio to the local drive, as shown in Fig. 2.10.

The user input panel has two modes: batch and stream. In the stream mode, the interface allows users to submit queries at a custom rate to the server database and update the visualization periodically. In the batch mode, a single query is submitted upon a button clicked by the user. A query contains various information about the desired acoustic events, which includes their timestamp, location, duration, frequency content, and label (tag), along with general authentication (i.e. the main server IP address, the name of a database, and a private key). Besides, to help with the data visualization, events can be ranked (by heatmap color codes across space or heights along time) based on any scalar, numerical record field (see Fig. 2.11).

2.3 Summary and conclusions

This chapter proposes a strategy for the integration of sensing and inference that is not only resource-efficient but can also support multiple applications. The strategy divides the sensing-inference process into multiple, smaller modules, where the inference result of the current module can be used to guide the execution of subsequent ones. This modular design has multiple advantages. First, it allows limited resources to be dynamically allocated where they generate the most value, thus improving the overall resource efficiency. Second, the system resource consumption can be further reduced by having multiple applications collaborate and share results of common modules.

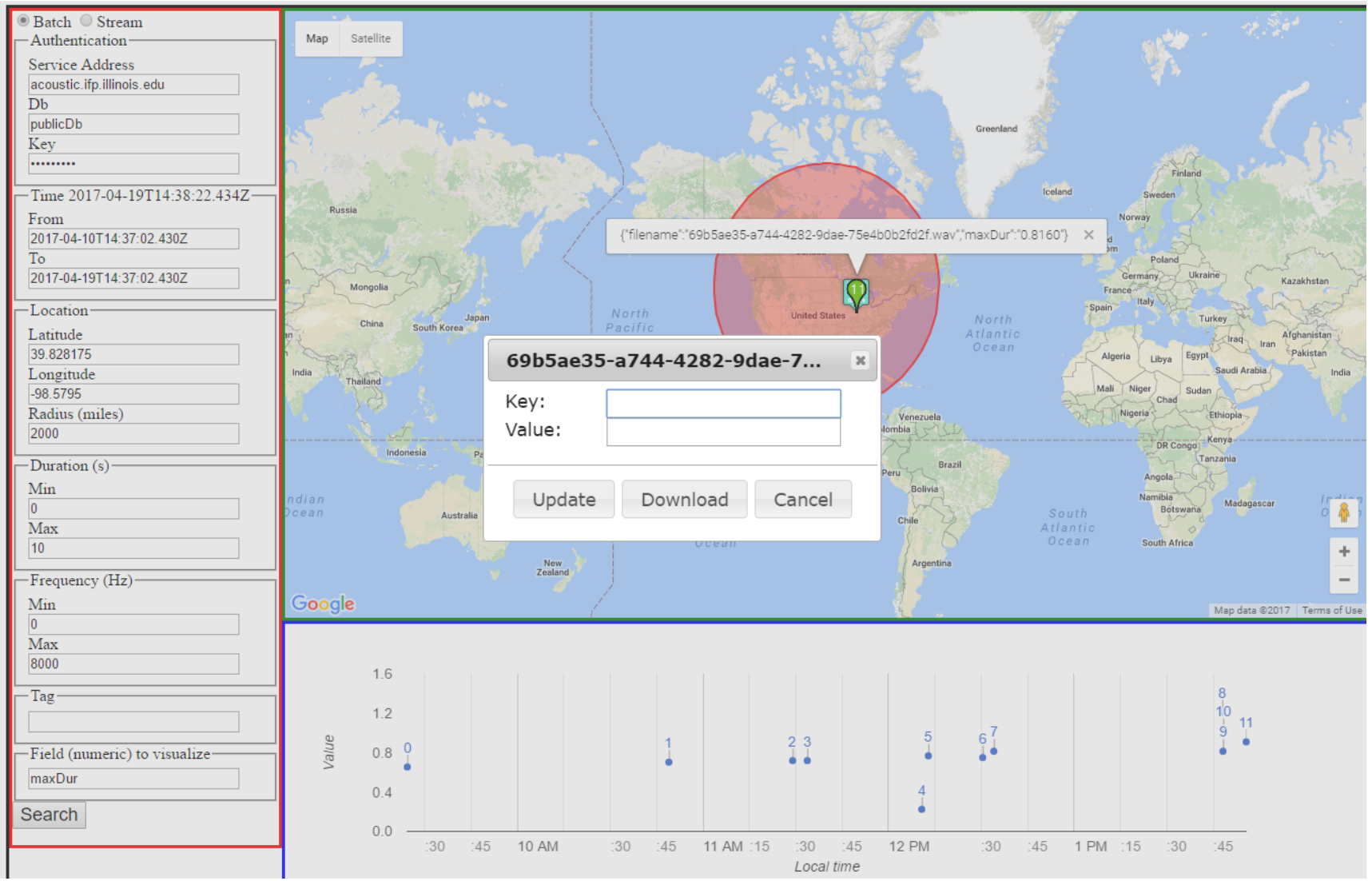


Figure 2.10: The graphical user interface (GUI) of the proposed system. Events can be manually tagged/labeled.

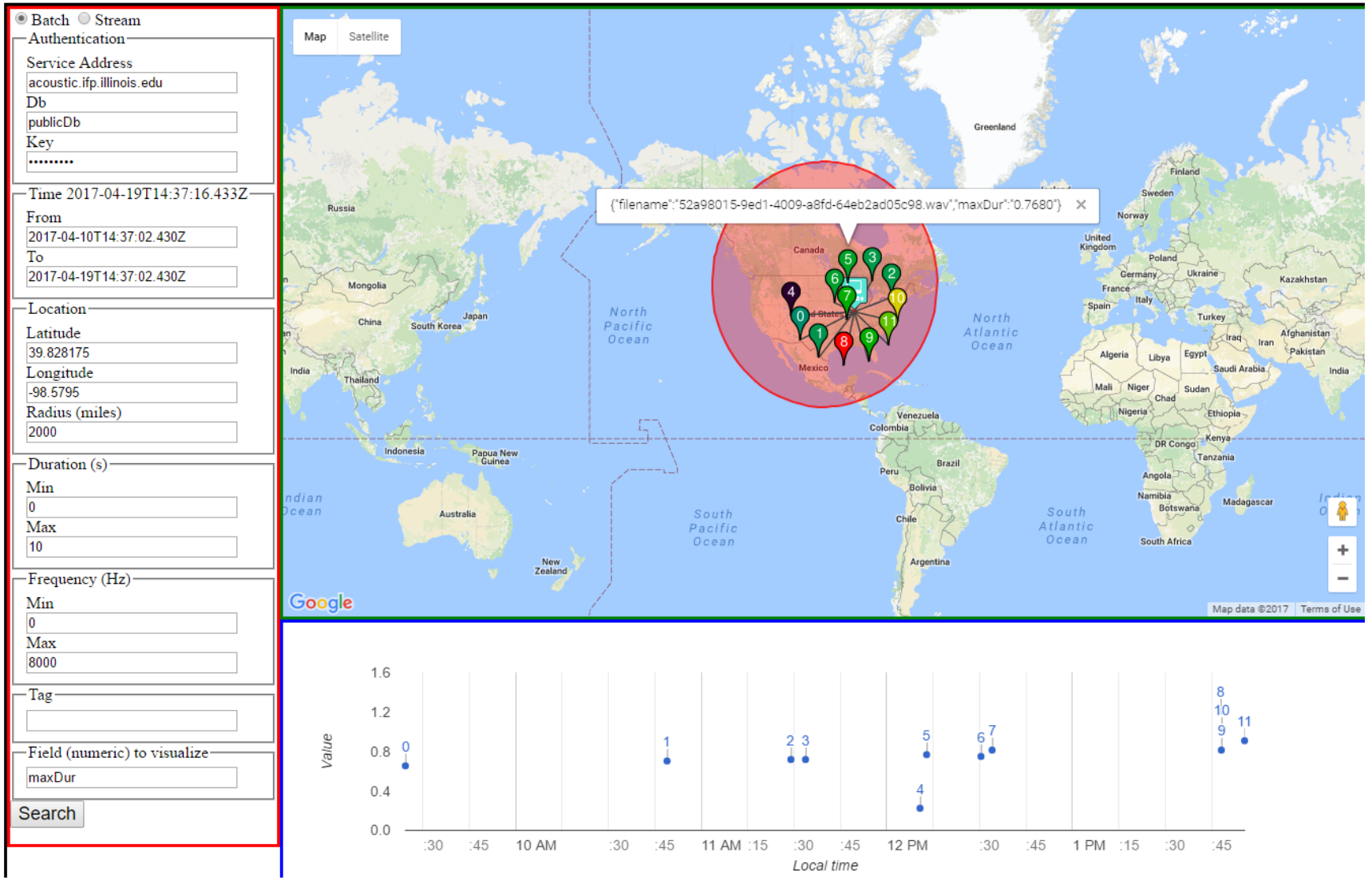


Figure 2.11: The system GUI visualizes acoustic events based on a numeric field set in the control (left) panel. In the spatial (upper right) panel, event values are color-coded, with red being the largest and black being the smallest. In the temporal (lower right) panel, event values are shown on the y-axis.

CHAPTER 3

BACKGROUND ON SENSING AND INFERENCE OPTIMIZATION

The modular system proposed in Section 2.2 is distributed by nature, and hence can benefit from the vast literature of distributed inference theory.

3.1 Distributed inference with quantization

Early work in distributed inference focuses on the problem of reliable detection under a bandwidth constraint on the communication between sensor nodes and a fusion center. A natural strategy to address this problem is quantization of observations. Tenney and Sandell [33] extended the classical centralized detection theory to the distributed setting. In [33], each sensor outputs a binary decision that jointly minimizes a global objective function. The decision rule at each sensor is still the well-known likelihood ratio, but the thresholds are coupled. Note that there was no explicit fusion center in [33]. Hoballah and Varshney [34] later introduced the fusion center into the distributed detection model and solved for both the fusion rule and the local decision rule at each sensor node by alternately optimizing one while keeping the other fixed, i.e. person-by-person optimization. Sensor nodes output binary decisions which become inputs to the fusion center. The Neyman-Pearson variant is covered in [35].

Gastpar and Vetterli [36] proposed an abstract model for WSN and raised the question of whether Shannon's separation theorem enjoyed by point-to-point communication still holds true for this model. It is well known that the separation of source and channel coding is optimal as long as there is overlap between the rate-distortion and the capacity-cost regions. However, this is not the case in general. In fact, [36] showed that, for the simplifying case of Gaussian WSN, even a simple joint source-channel coding scheme can result in a better achievable lower bound on distortion than the sepa-

rated one for the same total power. Specifically, using the separated coding scheme, the achievable distortion (defined to be mean squared estimation error) is lower bounded by $1/\log(M)$, where M is the number of nodes in the network, while the simple scheme of scaling an uncoded signal (to meet the power constraint) and employing a MMSE decoder (knowing that the signal is uncoded) yields a lower bound of $1/M$. Even though this result is limited to WSNs where all statistics involved are assumed to be Gaussian, in general, a joint approach does not perform worse. Further analyses in [37] show that the best achievable lower bound on the end-to-end distortion can be decomposed into two terms, one due to the noisy nature of the measurement process, and another due to the uncertainty in the communication channel. Interestingly, the distortion due to noisy measurements drops as $1/M$. To make the distortion due to communication also drop as $1/M$, the requirement on the communication bandwidth and power was identified as follows. If the number of sources equals the effective bandwidth, which is the product of temporal (the number of channel uses) and spatial (the number of receivers at the channel output) bandwidth, then the required power is constant. Otherwise, the required power grows exponentially with the number of sources. This result suggests that, for the same level of distortion, significant energy-saving can be achieved by providing adequate bandwidth.

Chamberland and Veeravalli [38] apply large-deviation theory to analyze the asymptotic performance (i.e. as the number of observations per sensor goes to infinity) of a wireless sensor network (WSN) under a communication constraint. By minimizing the Chernoff information (i.e. the error exponent) of the fusion rule, it was shown that, for Gaussian and exponentially distributed observations, a sensor network with identical *binary* decision rules is optimal. The fact that a binary decision rule outperforms other (e.g. quaternary) decision rules implies that the benefits of having more sensors outweighs that of having higher resolution per sensor reading. While the result also holds for the Neyman-Pearson criterion with independent Gaussian observations, it does not for general observation distributions.

The performance of power-constrained WSNs in the asymptotic regime, where the total power constraint tends to infinity, is studied in [39] by Chamberland and Veeravalli. It was shown that having sensors with the same transmission/quantization policy is asymptotically optimal. This result has profound implications, as it greatly simplifies the design of large

WSNs. Then, two candidate transmission policies can be compared using the normalized Chernoff information for a Bayesian formulation (the proof follows a Chernoff-bound [40] argument). The comparison criterion is instead the normalized relative entropy for a Neyman-Pearson formulation (Stein's lemma [40] is needed in the proof).

3.2 Distributed inference with censoring/control

Later work realized the significance of 1) the overhead cost in the usage of communication resources, i.e. communication links must be established using control packets before any data packets can be sent, and 2) the bursty nature of information-rich events in data streams [7]. This fundamentally changed the structure of the decision rule on sensor nodes: Instead of quantizing and sending every observation, a sensor node is constrained to only send complete observations occasionally. This is often referred to as the censoring approach, which is first proposed in [41] by Rago et al.

3.2.0.1 Open-loop strategy

Open-loop (offline) censoring can be formulated as a sensor-selection optimization as discussed in [42]. Joshi and Boyd [42] predetermined the optimal communication plan of a WSN by relaxing the knapsack-like sensor-selection constraint $\{0, 1\}$ to a convex one $[0, 1]$ and applying convex optimization. The convex solution, which takes on real values in the $[0, 1]$ interval, can be used to rank sensors and select only the top k ones. This optimization avoids the brute-force search with $\binom{m}{k}$ possible sensor combinations, where m is the total number of sensors in the network.

Joint quantization and censoring approaches are also found in the literature. Li and AlRegib [43] concentrate on the optimization of quantization levels for heterogeneous sensor nodes, which has direct consequence on the network's transmission power and estimation error. They optimize the quantization level to minimize the variance of the best linear unbiased estimator (BLUE) at the fusion center subject to a total power budget for transmission. Different optimal quantization levels at each heterogeneous sensor result in different MSE and power consumption. By ordering and selecting

only sensors with the smallest MSE that simultaneously satisfy the total power constraint, a subset of sensors is permitted to quantize and transmit their observations to the fusion center. It is demonstrated that the proposed method achieved a distortion that is closest to the theoretical lower bound. Note that the optimization in [43] happens offline, and the optimal subset of sensors, along with their corresponding quantization levels, is predetermined before the network's operation.

The censoring strategy has also been applied for energy-efficient data acquisition in the WSN database community. For instance, Deshpande et al. [44] proposed a probabilistic model-based approach that takes advantage of both the correlation in the sensor data and the diminishing returns in the confidence/accuracy of results when more data are collected. In particular, a multivariate Gaussian distribution on queriable attributes, e.g. temperature and voltage of a sensor node, was maintained by the query processor and used to service various types of probabilistic queries such as range queries and value queries. A temperature range query requires the computation of the probability that a temperature reading falls within an interval, which follows straightforwardly from the temperature marginalization of the multivariate distribution. Similarly, a value query is responded to with the mean of the temperature marginal. The confidence level also follows directly from the model. For a given query, the observation plan (i.e. which sensor to collect data from) is optimized to trade off between the data collection cost and the confidence of the query result. The technique in [44] is incorporated to improve TinyDB [45], a popular declarative query language for sensor network databases. Chu et al. [46] refined the model-based approach by exploiting the event-driven nature of many sensing applications. In [46], the base-station caches a copy of the probabilistic models of the sensor network. To reduce the communication cost in the network, most queries can be answered using the cached model without requesting sensor readings. Communication occurs only when the sensor network detects anomalous events and needs to update the upstream model.

3.2.0.2 Closed-loop strategy

Early work in closed-loop censoring started by considering a single sensor. For a fixed time horizon and limited channel uses, Imer and Basar [47] present

the optimal decision strategies for the sensor communication and the remote estimation. Both the sensor and estimator keep track of the remaining channel uses. If there are still channel uses, then the optimal estimator is the well-known mean and max of the posterior probability for the mean squared and 0-1 distortion criteria, respectively. Otherwise, if there are no more channel uses, then the estimator uses the prior instead. The optimal sensor's communication strategy is *sequential* and has the censoring structure, with the optimal censoring regions obtained by solving a dynamic program. Intuitively, the censoring region contains the least informative observations, and the sensor only sends informative observations to the estimator.

A limitation of [47] is that it only considers the estimation of i.i.d. random sequences. Nayyar et al. [48] consider the setting where a sensor can collect perfect Markovian measurements and harvest energy over time, but must communicate its data to the estimator through a noisy channel. The general strategy of [48] is to find the decision strategy for multiple agents (sensor, estimator) from the view of the agents, which have only the common information between them, which is the estimator in this case. Thus the estimator not only provides the estimate given all of the available measurements up to the current time, but also *prescribes* the next communication strategy for the sensor. A solution structure similar to [47] arises, with the optimal estimator minimizing the expected distortion with respect to the current belief distribution; the sensor communication/censoring strategy can be obtained by solving a dynamic program. Furthermore, [48] shows that if the distribution of the measurements is almost symmetric and unimodal for any energy level, then the communication strategy has a simple threshold structure. Specifically, a measurement is sent to the estimator only if it deviates sufficiently from the mode value.

Closed-loop censoring using sensor management algorithms has been extensively studied in recent years; a survey can be found in [49]. The common framework for many sensor-management algorithms is the partially observable Markov decision process (POMDP) [50, 51, 52]. In POMDP, it is well known that the optimal sensing strategy, in which no-sense (censoring) is a special case, must be solved using a dynamic-programming procedure (i.e. Bellman's equation) with a PSPACE-complete computational complexity [53]. This calls for approximation and relaxation approaches for solving POMDP. The Gaussian assumption [54] is often used to approximate the in-

tractable posterior with the mean and the variance. On a different note, when the unsensed/censored belief state remains unchanged over time, the optimal control policy (generally a function of the entire belief vector) decouples into Gittins indices, which is a collection of functions on belief elements [55, 56]. When mutual information is the objective of the sensing management, then Williams et al. showed that the performances of the greedy sensing strategies are guaranteed to be close to the optimal one (i.e. within 1/2 of the optimal sequential sensing strategy) thanks to a mathematical fact that the maximizer of a submodular objective function can be well-approximated using greedy algorithms [57].

A fundamental result in censoring sensor networks is given in [41] where Rago et al. showed that having sensors send their likelihood ratios when they do not fall in a single censored interval is optimal, since the consolidation of censoring intervals does not result in the loss of optimality. Appadwedula et al. [58] showed that the optimal sensor decision rules are independent, provided that the fusion center has its own observations. The proof of this claim relies on the established result in [41] and shows that setting the lower side of the interval to 0 also does not result in a loss of optimality. The assumption that the fusion center has its own observations holds in scenarios where sensor nodes take turns to be the fusion center. In addition, a robust formulation, a generalized likelihood ratio test approach, and a locally optimum formulation were proposed to handle the uncertainty in the observation distribution.

The results in [58] assume that observations are conditionally independent given the latent state. Instead, Abu-Romeh and Jones [59] exploited the correlations in observations via a modified generalized likelihood ratio test (mGLRT). In mGLRT, censored observations are estimated at the fusion center before a detection is made. The estimation is carried out under both hypotheses, using prior knowledge about the statistics of the uncensored observations. Abu-Romeh and Jones demonstrated that in many cases, mGLRT achieves a performance much closer to the centralized scheme than the optimal rule derived with the independence assumption. It is worth noting that the optimal fusion rule for censoring sensors with dependent observations is known [60], but is too complicated to be applicable in practice.

In [61], Blum proposes a distributed delayed-transmission scheme in which a sensor either sends its local log-likelihoods at a delay inversely proportional

to the magnitude of the log-likelihood, or stops (censors the rest) when told by the fusion center that a sufficient number of log-likelihoods have been received for the constrained estimation. The implicit assumption is that informative observations have large magnitudes in their log-likelihood functions. Whenever a log-likelihood from a sensor is received, the stopping condition is checked to determine with high probability whether the current ML parameter estimate using a subset of log-likelihoods is close to that using all log-likelihoods. Thanks to the delay transmission scheme, the contribution from later sensors is less significant than that from earlier ones, and in most cases this scheme reduces the amount of data needed to be transmitted to the fusion center. Braca et al. [62] later take the ordering approach to the extreme where it is shown that, for detection applications, only one transmission is sufficient to make the detector error arbitrarily small, as long as the network size is large.

It is generally agreed that only uninformative observations should be censored. In [63] by Msechu and Giannakis, observations are implicitly declared as informative (not censored) if they deviate sufficiently from the expected value, assuming Gaussian distributed observations. These observations are sent back to the fusion center, where the MAP estimation is carried out using gradient descent. The optimality of the gradient approach is guaranteed under mild conditions that make the optimization convex. Similar results are also obtained for ML estimation. Quantization is also considered to further reduce the bandwidth consumption for both ML and MAP estimations.

Zheng et al. [64] studied the sequential estimation problem, where an observation model similar to [63] is used, but the latent variable's statistic varies over time. With this setup, *informative* is explicitly quantified as the square of the innovation, which was shown to be equivalent to the KL divergence between the prior and the posterior state distributions, under appropriate conditions. The informative observations are sent to the fusion center where a particle filter is used to fuse the data and infer the target state. The particle filter uses a likelihood with both the uncensored observations and indicators of censored observations, which conveys additional information, to update the particle weights. It is demonstrated in [64] that the proposed particle filter yields better performance for the same bandwidth constraint compared to one that does not use the indicators and the random sensor-selection approach.

The asymptotic performance of censoring sensor networks was analyzed by Tay et al. in [65], in which they generalize the censoring results in [39] to include not only the transmission strategy, but also the measurement strategy. A similar result was obtained: in the Neyman-Pearson setting, sensor cooperation was shown to be unnecessary and sensors can independently use the same policy.

CHAPTER 4

THE GUIDED-PROCESSING PRINCIPLE

4.1 Overview

Cisco predicted that by 2020, there will be 50 billion Internet-connected devices, ushering in the Internet of Things (IoT) paradigm [20]. Many of these devices will be sensors that autonomously collect data about the physical world. Along with supporting infrastructure such as databases and data-analytics/inference engines, the resulting *sensing system* is projected to enable many novel data-driven applications. While the new paradigm has much potential, it also comes with challenges. Among them are the 'volume' and 'velocity' [66] of the data that need processing. It is becoming evident that the naive approach of stream-all-the-data-to-the-cloud is too costly in term of resources. And since energy is the most valuable resource in the post-Moore's-law era, it is the target of interest for this work.

A straightforward approach to reduce the energy consumption of a sensing system is *duty-cycling*, i.e. sensors are periodically turned off to reduce the amount of data that needs processing. While this approach does result in a low-power system, it does not necessarily yield an *energy-efficient* one, since the inference performance was completely ignored. An alternative approach is to have sensor nodes detect information-rich data instances from a data stream before uploading to the cloud for further processing. Unlike duty-cycling, this approach not only reduces the data-load, but also guides the downstream processing toward more quality data (hence the name *guided-processing*). For instance, data streams from audio sensors contain mostly

Copyright 2017 IEEE. Some of the material in this chapter has been reproduced, with permission, from: Long N. Le, Douglas L. Jones. "Guided-Processing Outperforms Duty-Cycling for Energy-Efficient Systems." IEEE Transactions on Circuits and Systems I, Special Issue on Circuits and Systems for the Internet of Things - From Sensing to Sensemaking, 2017.

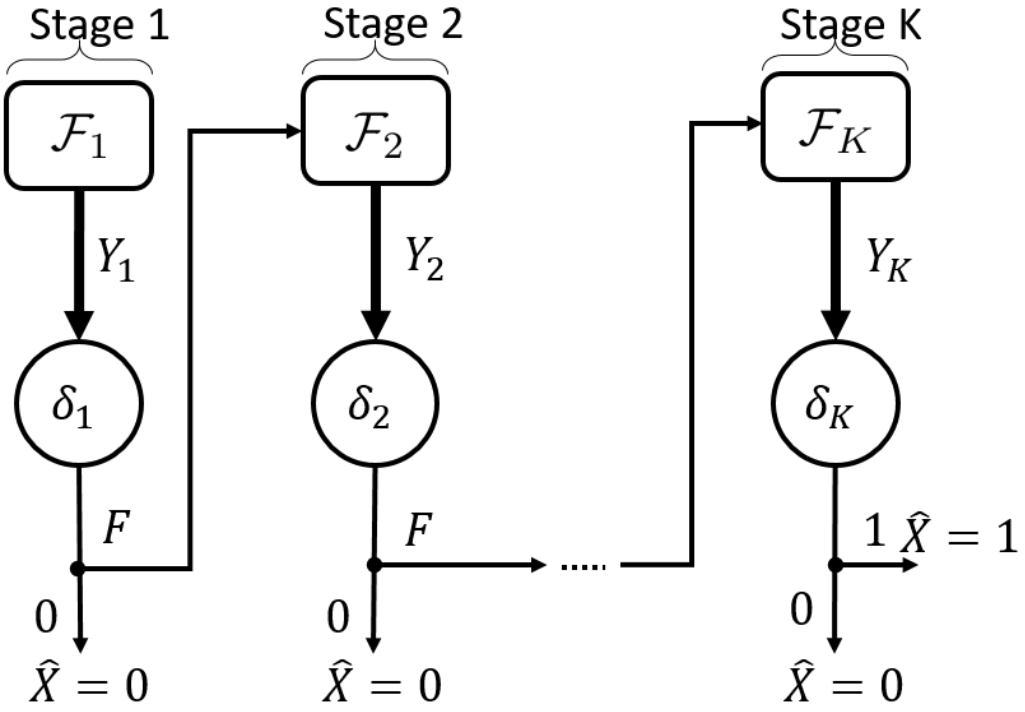


Figure 4.1: The cascade detection system with K stages (indexed by subscripts). For stage i, $\mathcal{F}_i$ denotes the feature extractor and δ_i denotes the binary decision of a detector. The feature itself is denoted by Y_i. X is the (detection) target's status, and $\hat{X}$ is the prediction about X by a detector.

background noise, which can be screened out early by sensors. Intuitively, guided-processing solves the issue of duty-cycling by explicitly accounting for the inference performance, *in addition* to the energy consumption.

An architecture of a detection system that implements the guided-processing approach can be visualized in Fig. 4.1. The system is a cascade of detection modules/detectors, each of which occupies a stage. A detector at stage i consists of a feature extractor $\mathcal{F}_i$, which produces the feature Y_i, and a decision rule δ_i, which takes Y_i and all previous features $Y_1, \ldots, Y_{i-1}$ as input. δ_i outputs different values depending on the stage (see Eq. (4.1)). X is the (detection) target state, which takes value 1 when the target is present, and 0 otherwise. Finally, $\hat{X}$ denotes the prediction of X by the detector.

The detection decision at each stage δ_i can take on the following values.

$$\delta_i = \begin{cases} 0: \text{ stop and declare } \hat{X} = 0 \text{ (negative)} \\ F: \text{ extract next feature} \end{cases}$$
$$i = 1, \ldots, K-1 \tag{4.1}$$
$$\delta_K = \begin{cases} 0: \text{ declare } \hat{X} = 0 \text{ (negative)} \\ 1: \text{ declare } \hat{X} = 1 \text{ (positive)} \end{cases}$$

Note that only negative decisions, i.e. $\hat{X} = 0$, are allowed at intermediate stages ($i = 1, \ldots, K-1$) since the goal is not to make the final decision (which is reserved for the last stage with the best performance) but to screen out early negative instances that are more likely in a rare-target setting.

The cascade architecture has been studied before in the literature. For instance, the seminal work by Viola and Jones [67] showed empirically that such a design is very effective in detecting rare targets in a large dataset (e.g. face detection), and was also proposed as a resource-efficient approach for stream mining by Turaga et al. in [68]. Detailed comparisons with existing works are articulated in Section 4.2. Our contributions here include the explicit modeling of performance uncertainties at intermediate layers/stages of a system. In addition, a realistic resource/energy consumption model is proposed. Finally, an in-depth comparison with the duty-cycling approach reveals a key insight on how a guided-processing system uniformly outperforms a duty-cycling one in term of energy-efficiency. Furthermore, it is worth noting that the proposed principle is more general than both the tandem/-cascade and the parallel (for instance, see [69]) structures, and can be applied to more sophisticated ones like trees and graphs to create inference-aware, low-power sensing systems (Chapter 5).

The rest of the chapter is organized as follows. Section 4.2 reviews prior works that studied the cascade structure, along with the limitations of their formulations/solutions. Section 4.3.1 sets up preliminaries for the system model presented in Section 4.3.2, where the method to optimize its operation is also discussed. The analytical comparison between the optimal cascade system and the duty-cycling one is given in Section 4.3.3. In Section 4.4, the proposed theory is applied to the design of an energy-efficient acoustic sensing system. Final remarks are given in Section 4.5.

4.2 Related works

It is worthwhile to note that the cascade detection system of interest here is different from the serial detector network in the distributed detection literature [70, 71, 72], in which the decision of a current module is treated as an extra observation, instead of as a control signal to *censor* subsequent modules and conserve resources.

The cascade architecture is prevalent in many inference applications, with the most widely-known example being the seminal work in face detection by Viola and Jones [67]. In [67], the system of cascaded detection modules is used to quickly discard many negative sub-images typically observed in face-detection applications, thus significantly speeding up the detection process. However, the cascade is not optimized in [67], leaving the optimal classifiers' parameters, both thresholds and weights, to be desired.

To this end, Luo [73] proposed to optimize thresholds of each detection module in a cascade using the classical Neyman-Pearson criterion, without consideration of resource cost. Under the assumption of statistical independence between detection modules, a gradient-based algorithm is proposed to search for the locally optimal thresholds, which is also a limitation of [73]. In contrast, our approach guarantees a globally optimal, resource-aware solution and does not assume independence between stages.

Ravindran et al. [74] proposed a cascade system comprising two stages of linear proximal support vector machines (SVMs) followed by one stage of radial basis function (RBF)-SVM and reported better generalization performance, compared to the RBF-SVM alone. Parameters of the classifiers were set heuristically so that upfront stages have low false-negative rate. However, there was no attempt at formalizing this rule of thumb and the resource/execution time improvement for using the cascade architecture was not studied.

Later, Jun and Jones [75] incorporated an energy resource constraint in the Neyman-Pearson-based optimization over thresholds of a two-stage cascade. In this setting, three solution types were identified: one that utilizes all of the available energy and false-alarm rate, one that utilizes all the energy while slacking the false-alarm constraint, and one that utilizes all the false alarm while slacking the energy constraint. An algorithm to find the optimal thresholds is only available if the true solution is of the first type. Later, it is proven in [76] that, if observations of the first stage are reused/resampled

in the second stage, then the first-type solution is optimal. Furthermore, the individual performance of the first and second stage detectors were used as the lower and upper bounds on the (detection) performance of the cascade, respectively. However, there was no comparison with the duty-cycling approach in term of energy-efficiency. Finally, unlike [75, 76], whose goal is the design of energy-efficient sensor nodes (for which a two-stage architecture is often sufficient), this work undertakes the design of an entire sensing system (for which there are likely more than one downstream processing). The new setting therefore motivates the development of a more general solution, i.e. cascade systems with an arbitrary number of stages.

Chen et al. [77] proposed a two-stage cascade architecture for an ultra low-power acoustic sensor. The first stage coarsely samples time-frequency (TF) characteristics of an audio stream and triggers the full TF analysis in the second stage if an acoustic event is detected. However, there was no attempt at optimizing the triggering threshold.

Chen et al. [78] designed a surveillance system using a two-stage cascade of low-end (acoustic and infrared) and high-quality (camera) sensors. The system in [78] can find a triggering threshold that either minimizes the detection error, or satisfies a constraint on the CPU utilization for video processing, but not both, and a heuristic was used to combine the two solutions: i.e., use the threshold that minimizes the detection error if it also satisfies the utilization constraint, otherwise use the one that satisfies the constraint. Unlike the ad-hoc approach of [78], our solution is derived from a well-defined framework. It is worth noting that Cohen [79] also studied a similar problem in which a multi-modal sensing system (with a PIR sensor and a camera) was designed for monitoring vehicles. While the treatment in [79] is principled (based on the partially-observable Markov decision process (POMDP) framework), the sensors are *not* operated in cascade, but instead are equally plausible options at each time step, and hence the approach is different from our work.

Since the optimization of the cascade is hard, Raykar et al. [80] relaxed the problem by assuming classifiers in the cascade produce soft/probabilistic outputs instead of hard decisions, and converted the joint optimization of classifiers' linear weights into a maximum *a posteriori* problem. Feature costs are also incorporated into the optimization using the standard Lagrangian argument, and a gradient-based algorithm is used to find the optimal weights. However, the thresholds must be found using an exhaustive grid search, which

is computationally intensive for cascades with many classifiers. Our solution does not suffer this drawback.

Chen et al. [81] proposed a cyclic optimization algorithm to optimize the linear weights of the classifiers in the cascade, along with their early-exit thresholds. That is, at each iteration, the algorithm cycles through all classifiers in the cascade, optimizing each one while leaving others untouched. The algorithm stops when the loss function no longer improves. A disadvantage of such an optimization procedure is that it requires multiple passes through the cascade, and there is no theoretical bound on the number of iterations it will take. In contrast, our solution requires only a single pass through the cascade.

In stream mining, Turaga et al. [68] employed a cascade of Gaussian mixture model (GMM)-based classifiers and formulated a problem to find both the number of mixture components and the threshold in each classifier that maximizes the system detection rate subject to constraints on false alarm, memory and CPU. The solution in [68] takes a person-by-person approach in which it iterates between 1) finding optimal numbers of mixture components, i.e. resource allocation, for all classifiers given thresholds, and 2) finding optimal thresholds for a given resource allocation. However, this approach failed to capture the direct dependence of the cascade's resource consumption on its thresholds, and is inherently suboptimal.

A limitation of the above works is that they only considered open-loop solutions where the thresholds are independent variables to be optimized. Ertin [82] considered closed-loop solutions for the two-stage cascade detection problem where the optimal decision rule at each stage, which is observation-dependent, is sought. It was shown that the optimal policies are still likelihood ratio tests, but with coupling thresholds, i.e. the threshold at a stage depends on the receiver operating characteristic (ROC) and the threshold of the other stage. Namely, the optimal thresholds cannot be found using the solution technique employed by [82]. Note that, unlike classical detection problems, optimizing thresholds in a cascade is critical in the trade-off between inference performance and resource cost. A contribution of this paper is finding the optimal parameters (both test-statistics and thresholds) for general detection systems.

Trapeznikov et al. studied a generalization of the cascade that was termed multi-stage sequential reject classifier (MSRC), which is simply the cascade

with an additional positive decision [83] or multiple additional (classification) decisions [84] at intermediate stages. Their resource-consumption model is 'nebulous', i.e. if the decision at an intermediate stage is to defer to the next stage, an abstract, *independent* "penalty" is incurred. In contrast, in our resource model, these penalties are shown to be precisely the Lagrangian-weighted feature extraction costs, and hence they are coupled (see Eq. (A.7)).

On the other hand, a resource-consumption model closely related to ours was considered by Wang et al. in [85]. The minor difference is that, instead of being proposed, our model was derived from first principles. However, [85] formulated the problem using the empirical risk minimization framework, since it was assumed that probabilistic models of high-dimensional features cannot be estimated. We take a different approach where it is assumed that probabilistic models of features *can* be estimated, by first reducing the features' dimensionality. In other words, the inputs into our algorithm are (probabilistic) models, not a dataset as in [85]. In addition, the solution proposed in [85] is a convex linear-program, which requires a convex relaxation (with an upper-bounding convex surrogate function) of the true objective function. In contrast, our solution is a dynamic program and requires no relaxation.

4.3 Optimality analysis of a cascade detection system

4.3.1 Feature models

For the rest of the chapter, the colon notation is used to denote a collection, e.g.

$$y_{1:i} \triangleq \{y_1, \ldots, y_{i-1}, y_i\} \tag{4.2}$$

Recall that Y_i denotes the feature used by the detector at stage i, and is modeled as a random variable whose distribution depends on the latent target $X \in \{0, 1\}$, i.e.

$$Y_i \sim \mathrm{p}_i(y_i|x), x \in \{0, 1\}, i = 1, \ldots, K \tag{4.3}$$

where lower-case letters denote realizations of the corresponding random variable in upper case and p denotes a probability mass/density function. It is

assumed that these distributions are *stationary* and hence can be estimated during training. While the stationary assumption might seem too restrictive at first glance, it does not preclude practical implementations of subsequent results, as will be shown in Proposition 4.1. Finally, it is worth noting that the feature (conditional) distributions in (4.3) are chosen by Nature and thus are *conditionally independent* of prior stages' decisions (if any), given the target state. The decisions do influence the *belief* about the latent state though.

Using Bayes' rule, the posterior probability of target presence is given by

$$\begin{aligned} \pi_1(y_1) &= \frac{1}{1+\frac{1-\pi_0}{l_1(y_1)\pi_0}} \\ \pi_i(y_{1:i}) &= \frac{1}{1+\frac{1-\pi_{i-1}(y_{1:i-1})}{l_i(y_i)\pi_{i-1}(y_{1:i-1})}} \\ & i = 2, \ldots, K \end{aligned} \tag{4.4}$$

where $l_i(y_i) \triangleq \mathrm{p}_i(y_i|1)/\mathrm{p}_i(y_i|0)$ and $\pi_i(y_{1:i}) \triangleq \mathrm{P}(X=1|y_{1:i})$ are the likelihood function and posterior probability at stage i, respectively. $\pi_0 \triangleq \mathrm{P}(X=1)$ is the prior probability of the target presence. Finally, the evidence probability is given by

$$\mathrm{p}_i(y_i|y_{1:i-1}) = \mathrm{p}_i(y_i|1)\pi_{i-1} + \mathrm{p}_i(y_i|0)(1-\pi_{i-1}) \tag{4.5}$$

It is worth noting that while the discussion herein only focuses on probabilistic and not computational models,[1] it does not preclude the application of the proposed framework for the latter. Indeed, a classical result by Richard and Lippmann shows that trained neural network classifiers, a popular class of computational model, also estimate Bayesian *a posteriori* probabilities [87].

An important aspect of the cascade detection system is that, except for the last stage, the main goal of other stages is to quickly screen out negative instances, and not to make the final decision. Therefore features used at stages other than the last one are suboptimal for the detection task by design, to keep the cost of their execution low. For instance, the all-band energy feature can neither characterize a bandpass target precisely, nor distinguish

[1] Terms coined by Bengio et al. in the context of feature/representation learning [86].

between a bandpass target and another bandpass interference, but can still be useful in the cascade thanks to its low cost [88]. The sub-optimality of these early-stage features, either due to 1) the failure to discriminate the target against potential interferences, or 2) the insufficient modeling of the target, can all be modeled as *uncertainty* in feature models. To this end, we employ the following *least-favorable* feature density models, developed by Huber in the context of robust detection [89],[90, Chapter 10],[91, Chapter 6], in place of the nominal ones.

$$\begin{aligned}
\mathrm{p}_i(y|0) \leftarrow & \begin{cases} \frac{1-\epsilon_{0i}}{v'+w'l_{Li}}[v'\mathrm{p}_i(y|0)+w'\mathrm{p}_i(y|1)], l_i(y) < l_{Li} \\ (1-\epsilon_{0i})\mathrm{p}_i(y|0), l_{Li} \le l_i(y) \le l_{Ui} \\ \frac{1-\epsilon_{0i}}{w''+v''l_{Ui}}[w''\mathrm{p}_i(y|0)+v''\mathrm{p}_i(y|1)], l_i(y) > l_{Ui} \end{cases} \\
\mathrm{p}_i(y|1) \leftarrow & \begin{cases} \frac{(1-\epsilon_{1i})l_{Li}}{v'+w'l_{Li}}[v'\mathrm{p}_i(y|0)+w'\mathrm{p}_i(y|1)], l_i(y) < l_{Li} \\ (1-\epsilon_{1i})\mathrm{p}_i(y|1), l_{Li} \le l_i(y) \le l_{Ui} \\ \frac{(1-\epsilon_{1i})l_{Ui}}{w''+v''l_{Ui}}[w''\mathrm{p}_i(y|0)+v''\mathrm{p}_i(y|1)], l_i(y) > l_{Ui} \end{cases} \\
& i = 1, \ldots, K-1
\end{aligned} \tag{4.6}$$

where the '$\leftarrow$' symbol is the assignment operator and

$$\begin{aligned}
v' = \frac{\epsilon_{1i}+\nu_{1i}}{1-\epsilon_{1i}}, v'' = \frac{\epsilon_{0i}+\nu_{0i}}{1-\epsilon_{0i}} \\
w' = \frac{\nu_{0i}}{1-\epsilon_{0i}}, w'' = \frac{\nu_{1i}}{1-\epsilon_{1i}}
\end{aligned} \tag{4.7}$$

and $0 \le \epsilon_{0i}, \epsilon_{1i}, \nu_{0i}, \nu_{1i} \le 1$ are uncertainty parameters of stage i. l_{Li} and l_{Ui} are the lower and upper bounds of the likelihood ratio at stage i, respectively, and can be found by solving the equations outlined in [91, Chapter 6]. More details are given in Appendix A.4. Note that since the new least-favorable densities result in a bounded likelihood function, the corresponding posterior probability is also bounded.

$$\pi_{Li} \triangleq \frac{1}{1+\frac{1-\pi_{i-1}}{l_{Li}\pi_{i-1}}} \le \pi_i(y_{1:i}) \le \pi_{Ui} \triangleq \frac{1}{1+\frac{1-\pi_{i-1}}{l_{Ui}\pi_{i-1}}} \tag{4.8}$$

4.3.2 System model and optimization

Optimizing the cascade system amounts to finding optimal decision rules $\delta_{1:K}$ that jointly minimize the proposed system's Bayes risk R_B of incorrect decisions subject to an expected system resource (e.g. energy) constraint e.

$$\begin{aligned} &\min_{\delta_{1:K}} R_B(\delta_{1:K}) \\ &s.t.\ E(\delta_{1:K}) \leq e \end{aligned} \tag{4.9}$$

where E is the expected system resource consumption. The Lagrangian technique can be used to convert the constrained optimization problem (4.9) into the following unconstrained, yet regularized, one:

$$\min_{\delta_{1:K}} R(\delta_{1:K}) \triangleq \lambda E + R_B \tag{4.10}$$

where the parameter λ, which depends on the resource constraint e, couples the resource consumptions of all stages together and R denotes the *system risk*, which is a measure of the combined detection performance and resource consumption. Hence, it is evident that a system with a lower system risk is more energy-efficient.

The Bayes risk R_B is given as follows.

$$R_B \triangleq \int \mathrm{p}(\mathrm{d}y_{1:S}) \Big\{ C_M \pi_S(y_{1:S}) \mathbb{I}(\delta_S = 0) + C_A (1 - \pi_S(y_{1:S})) \mathbb{I}(\delta_S = 1, S = K) \Big\} \tag{4.11}$$

where S denotes the (unknown) *stopping stage* where a decision is made and $\mathbb{I}()$ is the indicator function that is one if its argument is true and zero otherwise. The first and second terms are the miss and false-alarm risks, respectively. Note that false alarms are only incurred at the last stage, where positive decisions are allowed (see Fig. 4.1).

Eq. (4.11) can be broken down into multiple terms as follows.

$$\begin{aligned} R_B &= \int \mathrm{p}(\mathrm{d}y_{1:i}) \sum_{i=1}^{K} C_M \pi_i(y_{1:i}) \mathbb{I}(\delta_i = 0, S = i) + \\ &\quad \int \mathrm{p}(\mathrm{d}y_{1:K}) C_A (1 - \pi_K(y_{1:K})) \mathbb{I}(\delta_K = 1) \\ &= \sum_{i=1}^{K} \underbrace{\int \mathrm{p}(\mathrm{d}y_{1:i}) C_M \pi_i(y_{1:i}) \mathbb{I}(\delta_i = 0, S = i)}_{R_{i,M}} + \\ &\quad \underbrace{\int \mathrm{p}(\mathrm{d}y_{1:K}) C_A (1 - \pi_K(y_{1:K})) \mathbb{I}(\delta_K = 1)}_{R_{K,A}} \end{aligned} \tag{4.12}$$

since the arguments of $\mathbb{I}()$ are disjoint events. $R_{i,M}, i = 1, \ldots, K-1$ are the miss (false negative) risks due to early negative decisions at intermediate stages. $R_{K,M}, R_{K,A}$ are the miss and false-alarm (false positive) risks due to incorrect decisions at the last stage. Note that the system has no false-alarm risk at intermediate stages, since the cascade structure does not allow early positive decisions to be made. There are two reasons for this. First, to a dummy detector that flips a coin to make decisions, rare target makes it more likely to incur a false-alarm than a miss. Second, intermediate stages with model uncertainties are also likely to be fooled by interference to trigger a false-alarm. Altogether it is relatively safe to ignore early positive decision, since they are too unreliable. Proposition 4.2 later shows precisely when this ignorance is unharmful.

The expected resource consumption at stage i is the resource cost of feature extraction, denoted by D_i, weighted by the probability of that feature being selected by the previous stage. In addition, even when features are *not* extracted, real systems also incur a small, but non-zero, stand-by power consumption which is modeled by $d_i < D_i, i = 2, \ldots, K$. Hence,

$$E \triangleq D_1 + \sum_{i=1}^{K-1} \left[D_{i+1} \mathrm{P}(\delta_i = F) + d_{i+1} \mathrm{P}(\delta_i = 0) \right] \tag{4.13}$$

where D_1 is weighted by 1 because the first-stage feature is always extracted. Lastly, D_i and d_i can be measured in practice by profiling the feature-extraction process, and resource costs generally go up by an order of

magnitude[2] from one layer to another.

The solution to Problem (4.10) is given by the following theorem.

Theorem 4.1. *(The optimal decision rules for the cascade)*

$$\delta_i^*(\pi_i) = \begin{cases} 0, \pi_i(y_{1:i}) < \tau_i^* \\ F, \ \textit{else} \end{cases}$$
$$i = 1, \ldots, K-1 \tag{4.14}$$
$$\delta_K^*(\pi_K) = \begin{cases} 0, \pi_K(y_{1:K}) < \tau_K^* \\ 1, \ \textit{else} \end{cases}$$

where $\tau_i^* \in [\pi_{Li}, \pi_{Ui}]$ are the optimal thresholds at stage i.

Proof. See Appendix A.1. □

Equation (4.14) in Theorem 4.1 shows that the posterior probabilities of intermediate stages can be used to guide the execution of subsequent stages by thresholding them to decide whether to stop or extract more features in the next stage. The final stage has a standard detection rule, with the posterior probability being thresholded to make a prediction about the target state. The optimal threshold values $\{\tau_i^*\}$, which are critical in this trade-off between performance and resource cost, can be found using Algorithm 1 in Appendix A.5.

The solution offered by Theorem 4.1 requires the conversion of a feature Y into a posterior π, which can be difficult for practical implementations. To address this issue, an alternative, adaptive form of the solution, similar to the one proposed in [76, Eq. (14)], is given as follows.

[2]It is noteworthy that this exponential cost increase is similar to that considered by Poor in the context of quickest change detection [92], where it is shown that the optimal statistic is still the well-known accumulated likelihood product, but additionally weighted by the exponential base at each iteration.

Proposition 4.1. *(Adaptive implementation) Let*

$$\begin{aligned} \delta_i^*(y_i) &= \begin{cases} 0, y_i < \eta_i \\ F, \ else \end{cases} \\ i &= 1, \ldots, K-1 \\ \delta_K^*(y_K) &= \begin{cases} 0, y_K < \eta_K \\ 1, \ else \end{cases} \end{aligned} \tag{4.15}$$

where the adaptive thresholds η_i are updated as follows.

$$\eta_i \leftarrow \eta_i + \mu(\hat{q}_i - q_i), i = 1, \ldots, K \tag{4.16}$$

with

$$\begin{aligned} q_i &\triangleq \mathrm{P}(\pi_i \geq \tau_i^* | \pi_{i-1}), i = 1, \ldots, K-1 \\ q_K &\triangleq \mathrm{P}(\pi_K \geq \tau_K^* | \pi_{K-1}) \end{aligned} \tag{4.17}$$

being the activation probabilities and $\hat{q}_i, i = 1, \ldots, K$ are their runtime estimates. Finally, μ is the adaptation step size. Then (4.15) *is equivalent to* (4.14)*, provided that the features' likelihood ratios are monotonic.*

The advantage of the adaptive form in Proposition 4.1 is that it does not require runtime posterior evaluations, but instead K probability functions (of the prior π_{i-1} of stage i) $q_i, i = 1, \ldots, K$ that can be computed at training time. Intuitively, the thresholds η_i in this implementation are updated to ensure that (4.15) produces decisions with the same probability measure q_i as that of (4.14), consequently making them equivalent (assuming all features have monotonic likelihood ratios).

Given the above optimal decisions, the Corollary 4.1 quantifies the corresponding performance of the cascade system.

Corollary 4.1. *(Optimal performance of the cascade)*

$$R^*(\pi_0) \triangleq R(\delta_{1:K}^*, \pi_0) = V_0(\pi_0) \tag{4.18}$$

where $V_0(\pi_0)$ is the result of the following recursion

$$
\begin{aligned}
V_K(\pi_K) &\triangleq \min(\underbrace{C_M\pi_K}_{\textit{miss risk}}, \underbrace{C_A(1-\pi_K)}_{\textit{false-alarm risk}}), \pi_K \in [0,1] \\
V_i(\pi_i) &\triangleq \min(C_M\pi_i, \\
&\qquad \underbrace{\lambda(D_{i+1}-d_{i+1}) + \mathbb{E}[V_{i+1}(\pi_{i+1}(Y_{i+1},\pi_i))]}_{\textit{expected next-stage value function}}), \\
&\pi_i \in [\pi_{Li}, \pi_{Ui}], i = 1,\ldots,K-1 \\
V_0(\pi_0) &\triangleq \lambda\sum_{i=1}^{K} d_i + \lambda(D_1-d_1) + \mathbb{E}[V_1(\pi_1(Y_1,\pi_0))]
\end{aligned}
\tag{4.19}
$$

And the corresponding optimal thresholds are given by

$$
\begin{aligned}
\tau_K^* &= C_A/(C_A+C_M) \\
\tau_i^* &= \max(\pi_{Li}, \min(\pi_{Ui}, \\
&\qquad \min\{\pi_i : V_i(\pi_i) - C_M\pi_i < 0\})), \\
&\quad i = 1,\ldots,K-1
\end{aligned}
\tag{4.20}
$$

where C_M, C_A are the costs associated with miss and false-alarm decisions.

Corollary 4.1 shows that the optimal performance achieved by the system can be found using a recursive procedure. The procedure has K iterations, each corresponding to a stage in the system. Starting from the last stage K and proceeding backward to 0, the value function V is recursively updated (see (4.19)). The last-stage value function V_K is the minimum of the miss and false-alarm risk across π_K. An intermediate-stage value function $V_i, i = 1,\ldots,K-1$ is the minimum of the miss risk and the *expected next-stage* value function, which requires the probabilistic updates in Eq. (4.4),(4.5). The final value function V_0 is the minimal risk achievable by the system.

Once a value function is known, then the corresponding optimal threshold can be found using just arithmetic operations, i.e. comparing the value function with the miss risk. For the last stage K, the optimal threshold can be given in closed form. Note that the intermediate stages' thresholds are capped between the upper and lower bounds due to model uncertainty (see Section 4.3.1).

The discussion so far has been focusing on optimizing parameters of the cascade design. A natural next question is whether the constraints of the

cascade design can be relaxed to further improve performance. Namely, would introducing additional degrees of freedom, i.e. early positive decisions in intermediate stages, to the cascade *always* improve its performance? Intuitively, when model uncertainties of intermediate stages are accounted for (see Section 4.3.1), and it is known *a priori* that the target is rare, early positive decisions are likely to have higher risk and hence are discouraged. Therefore, introducing additional early positive decisions does *not* always improve the performance of the cascade. The precise conditions for which the cascade design itself is optimal are given by the following proposition.

Proposition 4.2. *(Optimality of the cascade design) With model uncertainty, introducing additional early positive decisions in intermediate stages of the cascade does not improve performance, when*

$$\underbrace{\max\{\pi_i : V_i(\pi_i) - C_A(1-\pi_i) < 0\}}_{\textit{optimal threshold for early positive decision}} > \pi_{Ui}, \qquad i = 1, \ldots, K-1 \tag{4.21}$$

Proof. See Appendix A.2. □

The left-hand side of (4.21) is the optimal threshold corresponding to an early positive decision. Namely, these additional decisions also have threshold-based optimal policies (see Appendix A.2), and a posterior probability *above* such a threshold shall trigger an early positive decision. If such a threshold is above the upper bound on the posterior probability at a stage, then its early positive decision is never selected, and hence does not affect the performance of the cascade.

4.3.3 Guided-processing vs duty-cycling

As alluded to in Section 4.1, duty-cycling is an alternative low-power design in which the system switches between the *on* and *off* modes. The duration for the on mode is determined by the duty-cycle factor $\rho \in [0, 1]$, with $\rho = 1$ being always on and $\rho = 0$ being always off. When off, the system completely misses out any potential events. However, when on, the system uses the best feature model, i.e. an equivalent of the cascade's last stage. Hence, the duty-cycling design can be viewed as the extreme version of the cascade without

intermediate layers. The (Bayes) detection risk and the resource consumption of a duty-cycled system is therefore given by

$$\begin{aligned} R_{\mathrm{dc},B} &= \rho(\underbrace{R_{\mathrm{dc},M} + R_{\mathrm{dc},A}}_{\text{risk in the on mode}}) + (1-\rho) \underbrace{C_M \pi_0}_{\substack{\text{miss risk in} \\ \text{the off mode}}} \\ E_{\mathrm{dc}} &= \rho D_{\mathrm{dc}} + (1-\rho) d_{\mathrm{dc}} \end{aligned} \tag{4.22}$$

where $R_{\mathrm{dc},M}$,$R_{\mathrm{dc},A}$ are the miss and false-alarm risks during the on mode, respectively. D_{dc}, d_{dc} are the resource consumptions in the on and off modes, respectively.

In general, $D_{\mathrm{dc}} \geq D_K$ and $d_{\mathrm{dc}} \geq d_K$ because they include not only the resource consumption of the last stage, but also additional overhead needed to get the data there. In addition, $R_{\mathrm{dc},B} \geq R_{K,B} \triangleq R_{K,M} + R_{K,A}$ (see Appendix A.3). Hence, the theoretically best duty-cycling system is the one in which the above bounds are met with equality.

Optimizing the duty-cycling design is straightforward since the detection risk and the resource consumption are decoupled. Hence the optimal decision rule does not affect the resource consumption, and ρ can be adjusted to meet a resource budget. While the duty-cycling design has the advantage of being simple, it can result in a lower energy-efficiency compared to the cascade design. Indeed, Proposition 4.3 shows that the optimal cascade design can outperform even the best duty-cycling design uniformly (across all $\rho \in [0,1]$, for a given π_0).

While the duty-cycling design has the advantage of being simple, it can result in a lower performance compared to the cascade design. Indeed, Proposition 4.3 shows that the optimal cascade design can outperform the optimal duty-cycling design uniformly (across all $\rho \in [0,1]$).

Proposition 4.3. *(Guided-processing vs duty-cycling) The optimal cascade design outperforms the best duty-cycling design uniformly (across all duty-cycle factor $\rho \in [0,1]$), provided that*

$$R^*(\pi_0) \leq C_M \pi_0 + \lambda d_K \tag{4.23}$$

and

$$\underbrace{\sum_{i=1}^{K-1} R^*_{i,M}(\pi_0)}_{\textit{intermediate-stages' miss risk}} \quad \leq \quad \underbrace{\lambda(D_K - e)}_{\textit{weighted resource saving}} \tag{4.24}$$

where

$$\sum_{i=1}^{K-1} R^*_{i,M}(\pi_0) \triangleq V_{0,M}(\pi_0) \tag{4.25}$$

and $V_{0,M}(\pi_0)$ *is the result of the following recursion*

$$\begin{aligned} V_{K,M}(\pi_K) &= 0, \pi_K \in [0,1] \\ V_{i,M}(\pi_i) &= \begin{cases} C_M \pi_i, & \pi_i \leq \tau_i^* \\ \mathbb{E}[V_{i+1,M}(Y_{i+1}, \pi_i)], & else \end{cases}, \\ \pi_i &\in [\pi_{Li}, \pi_{Ui}], i = 1, \ldots, K-1 \\ V_{0,M}(\pi_0) &= \mathbb{E}[V_{1,M}(Y_1, \pi_0)] \end{aligned} \tag{4.26}$$

Proof. See Appendix A.3. □

Equation (4.23) is simply a sanity check to ensure that the minimal risk of the proposed design must be lower than that of doing nothing. Equation (4.24) is more involved and it highlights the core differences between the proposed and duty-cycling approaches. In terms of detection performance, the guided-processing approach fundamentally incurs more miss risk (i.e. additional miss terms) due to the introduction of intermediate stages, i.e. the left-hand side of Eq. (4.24) and defined in Eq. (4.25), to reduce the energy consumption. Hence, the key insight is, as long as the achieved resource saving, i.e. the right-hand side of Eq. (4.24), is more than the additional miss risk incurred (for a given π_0), then the guided-processing design uniformly outperforms even the theoretically best duty-cycling one.

4.4 System prototype

This section applies the theory developed in Section 4.3 to design an energy-efficient audio sensing system.

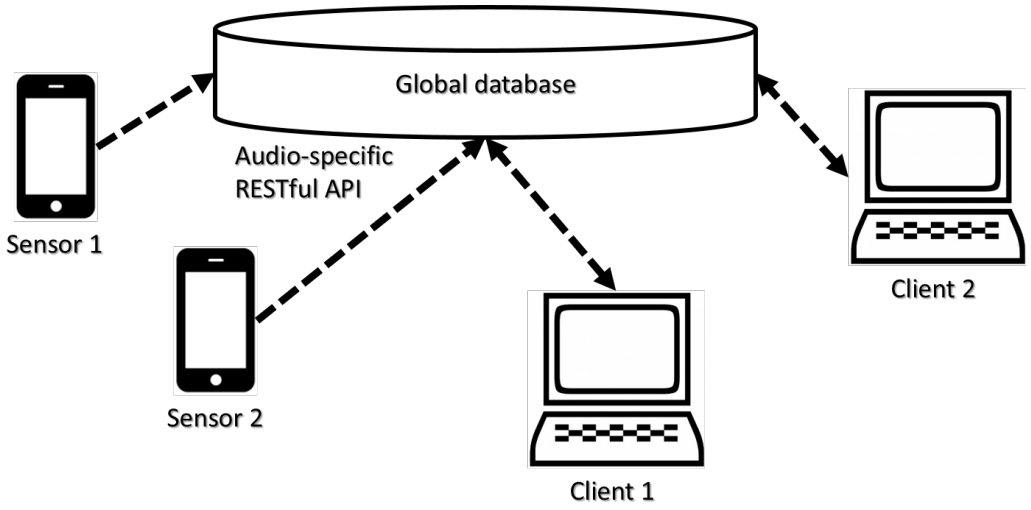


Figure 4.2: Devices of the prototype audio sensing system.

4.4.1 Hardware components

The proposed sensing system consists of three classes of devices: sensors, clients, and a globally-accessible data-plane [93] (see Fig. 4.2). In our current prototype, the data-plane is an instance of MongoDb database [28] with a custom RESTful interface specialized for audio data. Sensors are Android smartphones with our audio analysis app (see Fig. 4.3) installed.[3] Finally, clients are standard PCs running the Windows OS. The power consumption of sensors, profiled using Trepn [94] on a Nexus-5X, and clients, measured using *powercfg* on a 2.00 GHz machine, at different operating modes are listed in in Table 4.1.

Table 4.1: Power consumption at different modes of devices of the acoustic sensing system.

Devices & Modes	**Power consumption (mW)**
Android processing	84.36
Android transmission	1097
PC sleep	264
PC processing	15131

[3]Available for download at `https://play.google.com/store/apps/details?id=com.longle1.spectrogram`

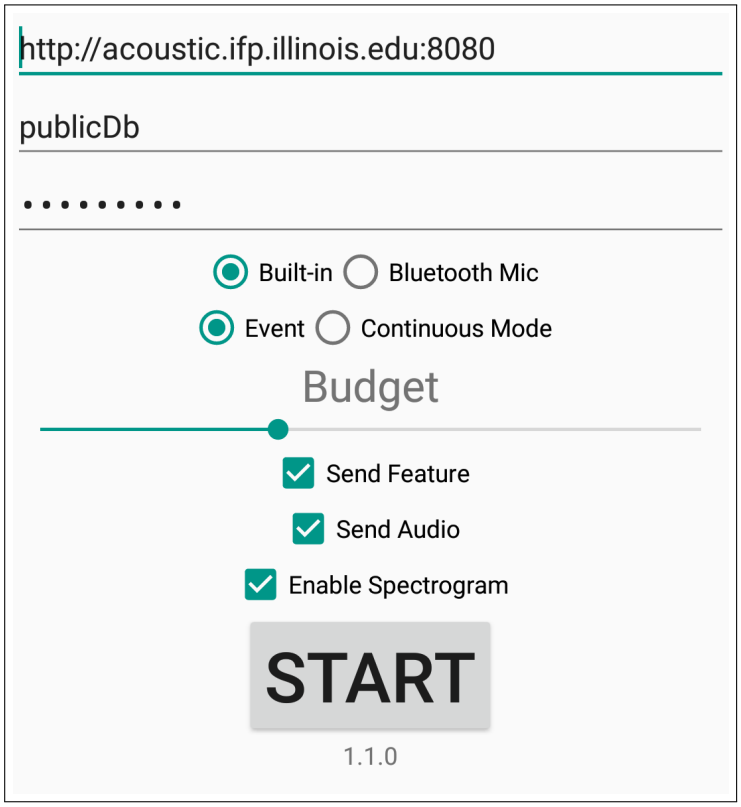


Figure 4.3: A screenshot of the Android-based audio analysis app. The app uses the adaptive implementation outlined by Proposition 4.1, with the probability q_1 input via the "Budget" slider.

4.4.2 Software components

While the proposed sensing system can be used for many applications, the detection of the Golden-cheeked Warbler (GCW)'s (type-A) calls [95] is chosen here as the application of interest. Namely, $X = 1$ indicates the presence of a GCW call, and $X = 0$ otherwise. Since the GCW is an endangered bird species, this application has important implications for their conservation.

The application's software is organized into three subtasks: generic energy-based analysis, spectral-based analysis, and temporal-spectral-based analysis. The energy analysis is a low-complexity computation that produces energy-based features useful for detecting acoustic events from silence. The spectral-based analysis takes into account the spectral information about the GCW calls, which only has energy in the 4500-6500 Hz and 7000-8000 Hz bands (see Fig. 4.4), to produce band-specific, energy-based features using standard DSP filtering techniques. Finally, the spectral-temporal-based analysis takes into account both the spectral and temporal structure of the GCW call from Fig. 4.4 to produce reliable, indicative features using a template-matching technique. Note that the input into the above analyses is an audio stream (or precisely, its high-dimensional time-frequency representation, see Fig. 4.4), and their output is a scalar score sequence, i.e. a score for each audio frame. Hence, these analyses effectively perform dimensionality reduction.

Since the generic energy analysis has low computational complexity and can help prune out a significant amount of noise-only data from the audio stream early, it is executed on edge/sensor nodes. Only *acoustic events* are transmitted downstream to clients, where spectral and temporal-spectral-based analyses are further carried out. The system diagram is illustrated in Fig. 4.5 and arranged to fit the proposed cascade abstraction. Note that the physical separation (between sensors and clients) does not necessarily correspond to the logical separation (between stages). For instance, the cost of data transmission on sensors is included into the cost of executing the second stage, along with the cost of spectral-based analysis on clients, since they are both a result of the first-stage decision.

The resource cost parameters at each stage $D_i, i = 1, 2, 3$, which can be estimated from values of Table 4.1 and the execution times of the software components, are needed to optimize the resource/performance trade-off. It is assumed that all processing finishes before a periodic deadline, i.e. when

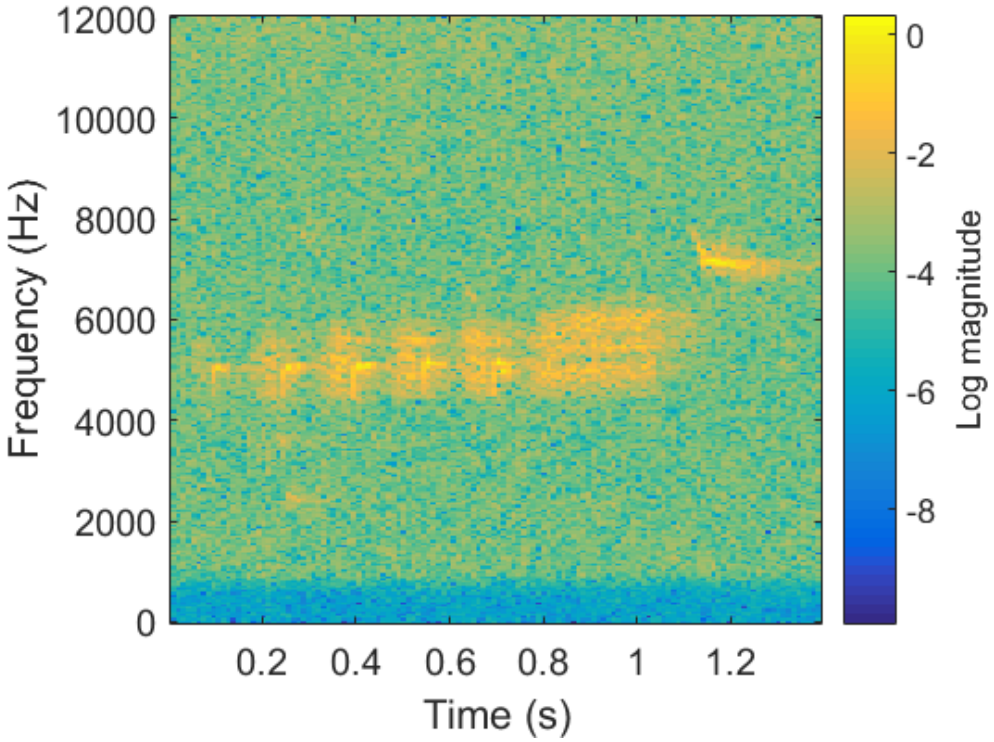


Figure 4.4: Spectrogram of a sample GCW's (type-A) call.

buffers (an ADC buffer on the sensor, a task buffer on the client) are full. The average execution time of each task (per audio frame of 32 ms) can be estimated/profiled and is given as follows. The energy analysis takes 16 ms.[4] The average transmission time is 11 ms (500 ms for a 1.5 s event[5]). Finally, the spectral and temporal-spectral analyses take 0.34 μs and 14 ms, respectively.[6] Hence,

$$\begin{aligned} D_1 &= 84.36 \times 0.016 \\ D_2 &= 1097 \times 0.011 + 15131 \times 0.34 \times 10^{-6} \\ D_3 &= 15131 \times 0.014 \end{aligned} \tag{4.27}$$

The off-mode/idle energy costs (per audio frame) on the client are given as follows.

$$\begin{aligned} d_2 &= 264 \times 0.34 \times 10^{-6} \\ d_3 &= 264 \times 0.014 \end{aligned} \tag{4.28}$$

[4]Estimated as half of the frame length.
[5]Profiled on the Android prototype.
[6]Profiled in MatLab on the PC.

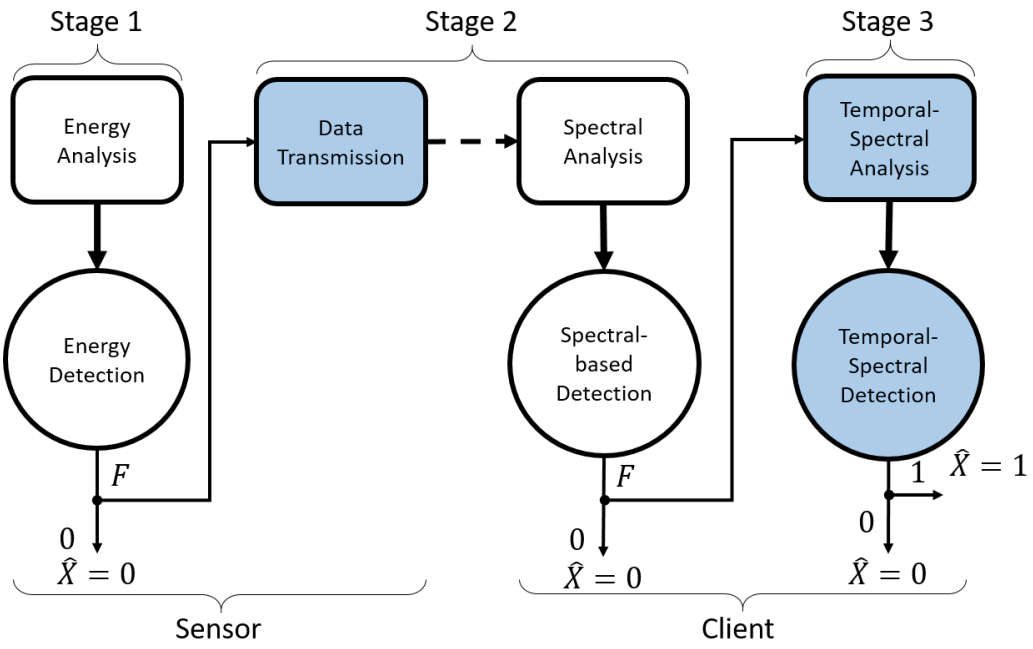


Figure 4.5: The software block diagram is organized as a cascade with 3 stages: energy analysis as Stage 1, spectral-based analysis (along with the data transmission) as Stage 2, and temporal-spectral analysis as Stage 3. Note that components of the cascade are distributed across the network, with the dashed line representing a remote connection. For comparison, a system with the duty-cycling design only has highlighted components, i.e. data transmission from sensor to a client where the temporal-spectral analysis is carried out.

The same system designed with the duty-cycling approach will have fewer components (only those highlighted in Fig. 6.3) and its resource/energy consumptions parameters are given as follows.

$$\begin{aligned} D_{\text{dc}} &= 1097 \times 0.011 + 15131 \times 0.014 \\ d_{\text{dc}} &= 264 \times 0.014 \end{aligned} \tag{4.29}$$

Our dataset is a 46-minute, 24 kHz audio field recording in Rancho Diana, San Antonio's city park. The dataset contains 206 GCW calls (manually identified and labeled), the duration of each being approximately one second. In addition to GCW calls, the dataset also contains various interferences from other animals' vocalization, time-varying wind noise, etc., since it is taken directly from a field recording. Precisely, the fraction of GCW calls in the entire dataset is 10.19%. Hence, this detection problem belongs to the rare-

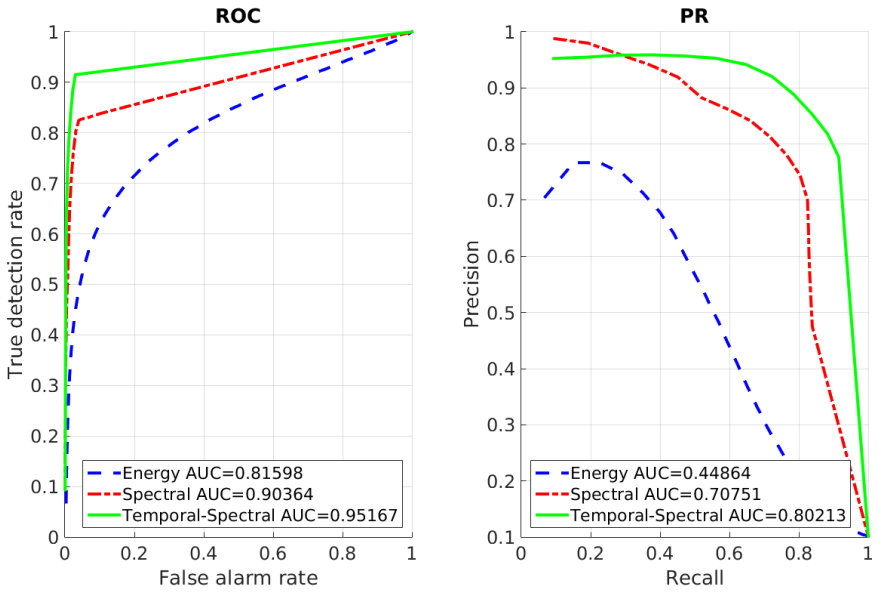


Figure 4.6: Receiver operating characteristic (ROC) curves and precision-recall (PR) curves of the features produced by the 3 analyses. Note that the dip in precision of the energy analysis (near the low recall part of the curve) is evident that the energy feature can be misleading, i.e. not proportional to the true (but unknown) likelihood ratio, since the precision curve is not monotonically increasing with a threshold. This evidence also supports the claim in Section 4.3.1 about uncertainty in feature models.

target class, where the prior is asymmetrical, i.e. $\pi_0 \ll 0.5$. Throughout this section, we consider a range of priors in the rare-event regime, i.e. $\pi_0 \in [0.05, 0.15]$. Finally, the miss and false-alarm costs are given by $C_M = 3, C_A = 1$ to emphasize that the miss risk is higher in this setting.

The data are input to each of the three analyses discussed above. The scalar output scores from each analysis are taken as its respective features, resulting in a total of three feature sets/groups/types. The discriminative power of each feature type, or equivalently the performance of an analysis, can be quantified using receiver operating characteristic (ROC) and precision-recall (PR) curves as shown in Fig. 4.6. From the figure, it is evident that the temporal-spectral feature is better than the spectral feature, which in turn is better than the generic energy feature, at detecting GCW calls.

The conditional probability mass functions (PMF), i.e. $\mathrm{p}_i(y_i|x)$, of features from each analysis can be estimated up to some quantization level, i.e. 100.

Furthermore, as alluded to in Section 4.3.1, energy-based and spectral-based features, by construction, are inadequate to characterize GCW calls, and hence there are inherent uncertainties in these features for the detection of GCW calls. These uncertainties can be explicitly accounted for in the features' distributions using the uncertainty model discussed in Section 4.3.1, with the following parameters.

$$\begin{aligned} \epsilon_{01} &= \epsilon_{02} = 0.1 \\ \epsilon_{11} &= \epsilon_{12} = 0.1 \\ \nu_{01} &= \nu_{02} = 0.1 \\ \nu_{11} &= \nu_{12} = 0.1 \end{aligned} \tag{4.30}$$

Intuitively, the ϵ and the ν parameters indicate the level and the strength of a contamination on the nominal distribution, respectively. A formal method to set these parameters is left for future work. Finally, it is assumed that the temporal-spectral analysis (the last stage) is sufficient to characterize GCW calls and hence there is no uncertainty in this feature set.

4.4.3 Results

The optimal thresholds/strategies for detectors in the cascade are given in Fig. 4.7. The equivalent, implementation-friendly version of the solution, as discussed in Proposition 4.1, is given in Fig. 4.8. Note that the decision functions of intermediate layers have limited supports due to model uncertainties. Furthermore, Proposition 4.2 can be applied to verify that there is no gain from having additional early positive decisions in this system.

The optimized system risk is further broken down into the weighted resource consumption, the miss and false-alarm rates in Fig. 4.9 to provide an intuitive understanding of the optimal policies.

The guided-processing system is compared against both the theoretically-best (ideal) and the real, energy-equivalent duty-cycling designs, to be defined herein. In the ideal case, the lower bounds on energy costs and detection risks are assumed to hold, i.e.

$$\begin{aligned} D_{\mathrm{dc}} &= D_K, d_{\mathrm{dc}} = d_K \\ R_{\mathrm{dc},M} &= R_{K,M}, R_{\mathrm{dc},A} = R_{K,A} \end{aligned} \tag{4.31}$$

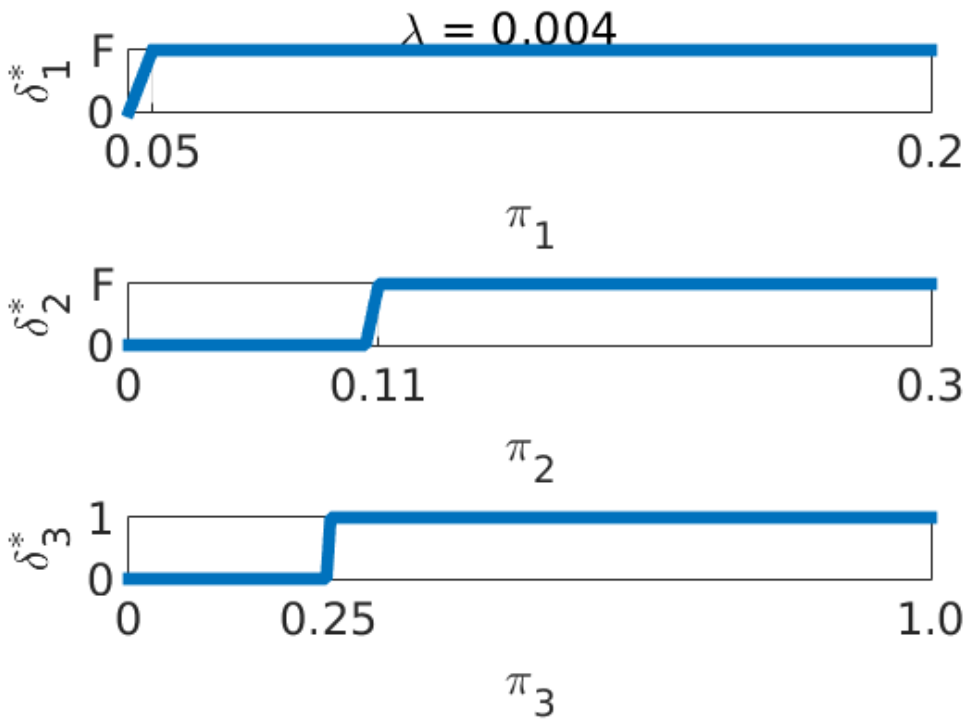


Figure 4.7: Optimal decision rules of the cascade system $\delta_i^*(\pi_i) \in \{F, 0, 1\}, i = 1, \ldots, 3$.

while the corresponding values in the real duty-cycling system must be measured directly. In addition, unlike the ideal case where it is sufficient to compare against $\rho \in \{1, 0\}$ (either completely on or off, see Appendix A.3), ρ must be adjusted in the real duty-cycling system to yield an equivalent energy consumption to the proposed one, thus allowing the two to be compared in term of their detection performance.

Furthermore, to demonstrate the generalization power of the proposed approach over that of [76], the system is also compared against its 2-stage version, where the spectral analysis in Fig. 6.3 is removed (i.e. the client only executes the temporal-spectral analysis instead of a cascade of it and a spectral-analysis).

The comparisons between the five approaches in terms of system risk (energy-inefficiency), energy consumption, false-alarm and miss rates are given in Figs. 4.10, 4.11, 4.12, 4.13, respectively. From Fig. 4.10, it is evident that the proposed approach is the most energy-efficient one (with the smallest system risk) across the prior π_0 of interest. Moreover, the ideal

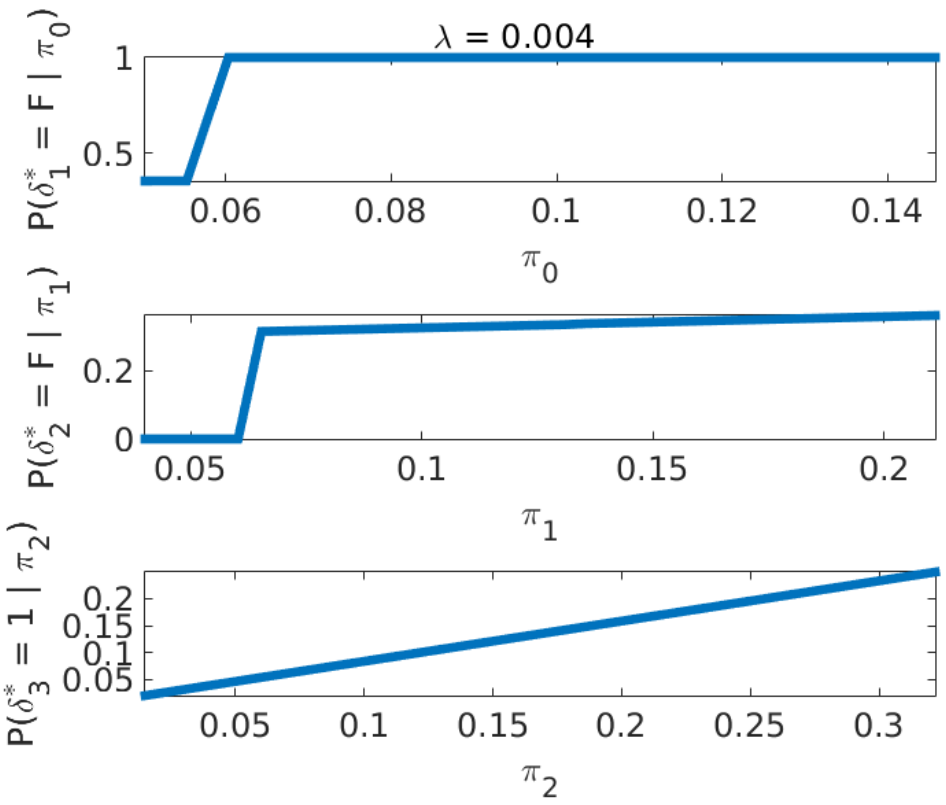


Figure 4.8: The alternative representation of the optimal solution for adaptive implementation.

bounds are tights, and generalization from two to three stages helps improve the overall energy-efficiency. Figures 4.11, 4.12, 4.13 together show that the guided-processing approach is able to stay between the two ideal bounds for all three metrics and outperform the real duty-cycling approach in both false-alarm rate (up to 1.7×) and miss rate (up to 4×) for the same energy consumption. Finally, it is worth noting that the removal of the spectral analysis module (resulting in the 2-stage version) strongly limits the design space and increases the total miss rate (even with one less miss term, see Fig. 4.13) consistently across priors, when compared to the proposed 3-stage system.

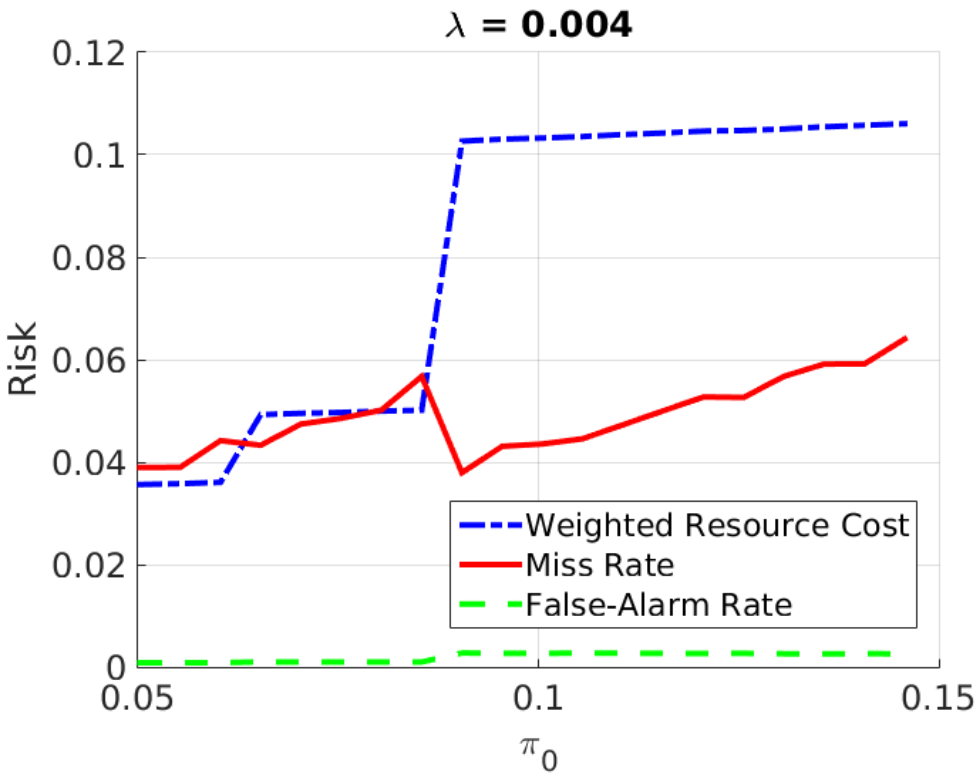


Figure 4.9: Breakdown of the system risk into components (see Eq. (4.10)): false negative (miss), false positive (false-alarm), and Lagrangian-weighted resource consumption. Low false-alarm rate is achieved across the priors of interest. The miss rate tends to increase with the prior. At a certain level, the system must ramp up its resource consumption or incur more false-alarm to reduce the miss rate.

4.5 Summary and conclusions

This chapter proposes the guided-processing approach for sensing system design and shows that it can be fundamentally more energy-efficient than the naive approach of duty-cycling. Empirical evidence from a practical application also supports the analysis. The proposed design was applied to develop an acoustic sensing service on top of which many applications can be built. These are publicly available online[7] for demonstration.

An apparent drawback of the proposed approach is its stationarity assumption and, as a result, the feedforward structure of the solution; i.e. the decision to invoke downstream processing, rests entirely on an upstream detector with a fixed policy. It is conjectured that higher energy-efficiency

[7] At `http://acoustic.ifp.illinois.edu`

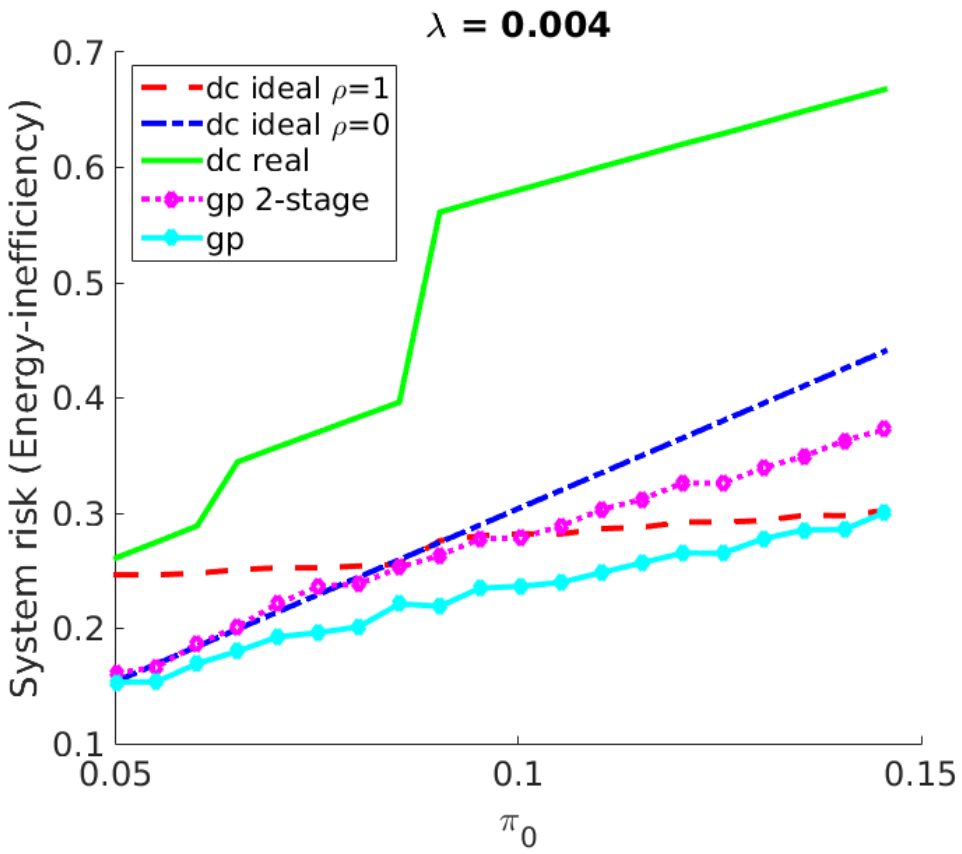


Figure 4.10: Comparison of system risk between the guided-processing (gp) and various duty-cycling (dc) approaches.

can be achieved by exploiting the temporal structure in extracted features, for which a feedback-based solution might arise. For instance, it is natural for downstream results to influence upstream policies/decision-making over time.

Another useful extension of the current energy-aware framework is the incorporation of a detection delay constraint, which is common for real-time applications. It is envisioned that the the solution policy is also time-dependent, but in response to a delay penalty rather than the temporal structure of features.

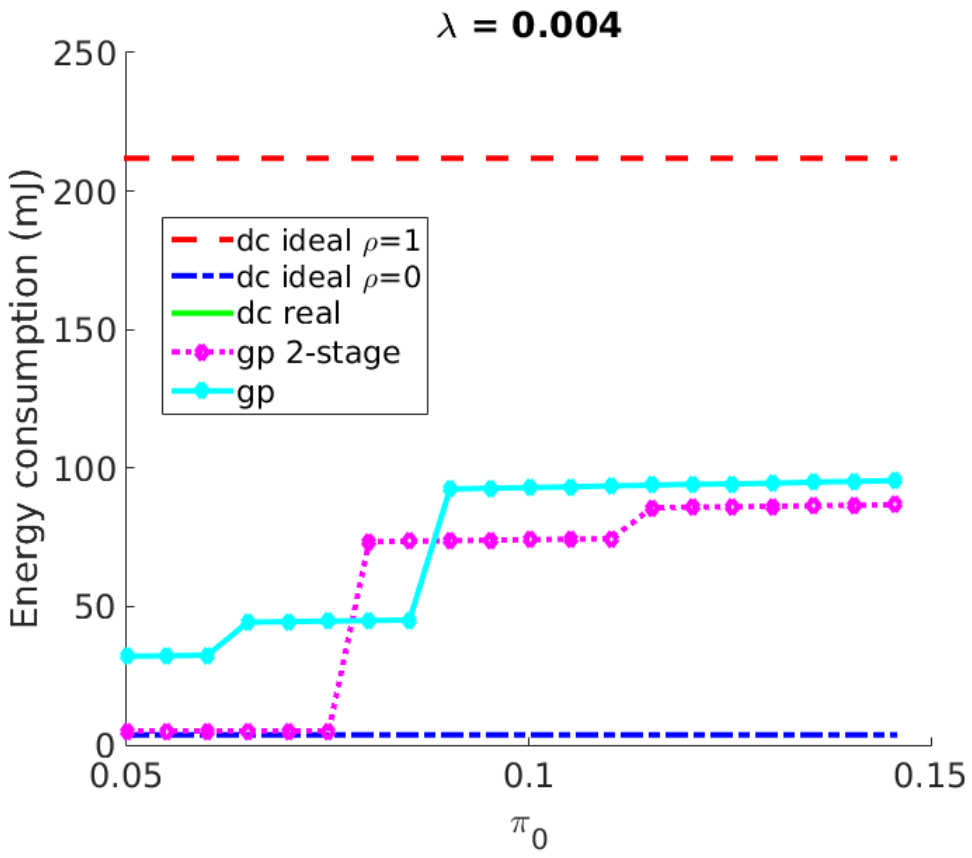


Figure 4.11: Comparison of energy consumption (per audio frame) between the guided-processing (gp) and various duty-cycling (dc) approaches. Note that the energy consumption of the real dc and gp approaches are the same by construction (i.e. their curves overlap by setting ρ appropriately).

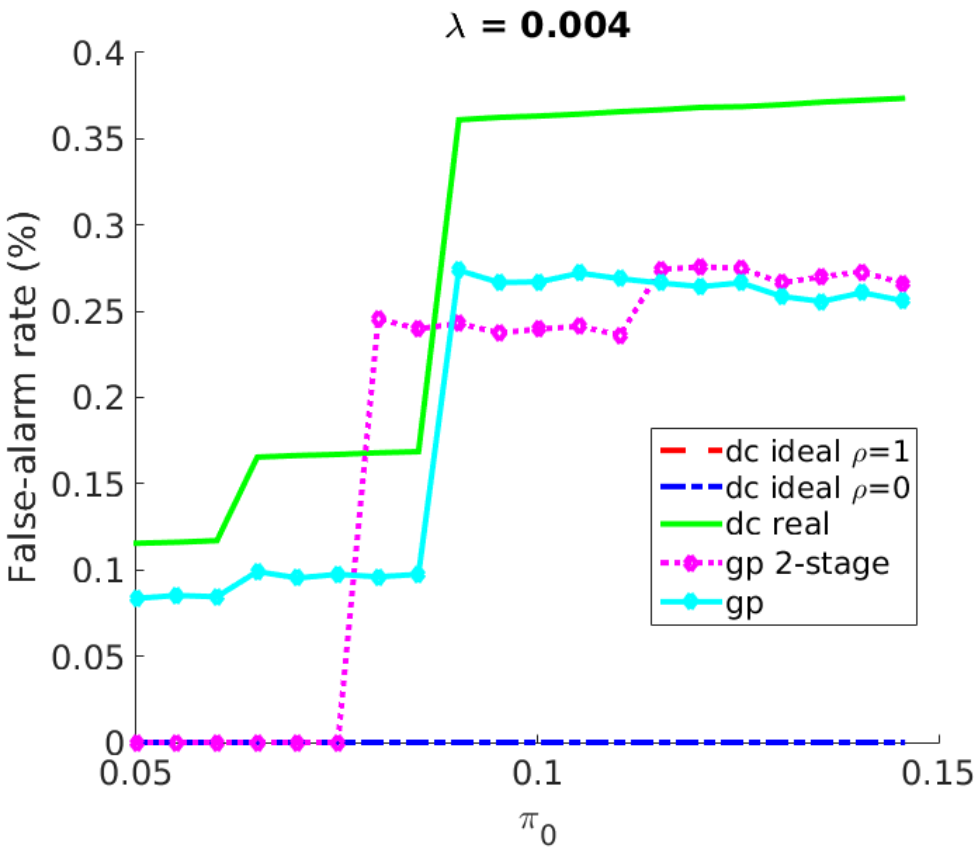


Figure 4.12: Comparison of false-alarm rate between the guided-processing (gp) and various duty-cycling (dc) approaches. Compared to dc real across π_0, gp is up to 1.7× lower in false-alarm rate. Note that gp and dc ideal $\rho = 1$ are overlapped.

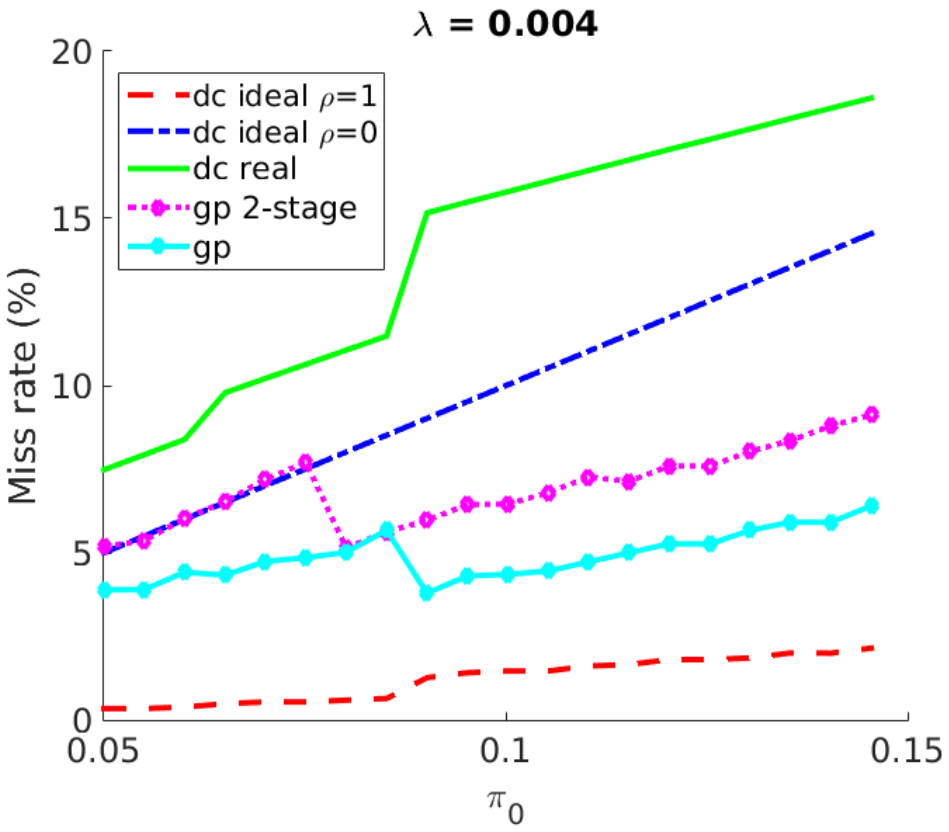


Figure 4.13: Comparison of miss rate between the guided-processing (gp) and various duty-cycling (dc) approaches. Compared to dc real across π_0, gp is up to 4× lower in miss rate.

CHAPTER 5

GUIDED-PROCESSING ON GRAPHS

5.1 Overview

The discussion of optimal resource management so far, i.e. Chapter 4, has been limited to systems with the cascade/tandem structure. While there exist generalizations in the literature that add more inference decisions, i.e. multi-nary classifiers [84, 85], there has been no generalization that adds more choice for downstream neighbors, i.e goes beyond the cascade structure. However, as alluded to earlier, the proposed guided-processing principle is general and can be applied on more sophisticated systems with tree- or (directed) graph-based structures, i.e. each module can have multiple downstream neighbors.[1] Given a tree/graph-based system whose nodes are detectors with different resource-performance trade-offs, the goal of guided-processing herein is to seek the set of operational policies that jointly optimize the total system's trade-off.

The remainder of this chapter is organized as follows. The guided-processing algorithm for a graph-based system is discussed in Section 5.2, followed by a system simulation in Section 5.3. Finally, conclusions are reached in Section 5.4.

Copyright 2017 IEEE. Some of the material in this chapter has been reproduced, with permission, from: Long N. Le, Douglas L. Jones. "Guided-Processing Outperforms Duty-Cycling for Energy-Efficient Systems." IEEE Transactions on Circuits and Systems I, Special Issue on Circuits and Systems for the Internet of Things - From Sensing to Sensemaking, 2017.

[1] It is worth noting that this generalization is analogous to that of linked lists to trees/-graphs.

5.2 Graph-based guided-processing algorithm

Recall from the discussion on cascade structures (Chapter 4) that the guided-processing solution starts from the last detector and works backward to the first one, since each stage depends on the value function of a downstream stage. Therefore, to adapt the established solution to graph-based systems, a *post-order traversal* through nodes is required, since each node, which herein represents a detection module, depends on value functions of its downstream neighbors. An obvious technical requirement is that there must be no cycle in the system's (directed) graph, i.e. only directed-acyclic graphs (DAG) are admissible. For instance, a post-order traversal on the graph in Fig. 5.1 is $10000 \rightarrow 1000 \rightarrow 1050 \rightarrow 1200 \rightarrow 100 \rightarrow 105 \rightarrow 110 \rightarrow 130 \rightarrow 10$. Note that while there are multiple valid post-order traversals, they are all equivalent from a guided-processing perspective.

Listing 5.1: Post-order graph traversal with a custom callback in Python

```
1  def postTraverse(nodes,start,cb):
2    # non-recursive/iterative implementation
3    # post-order traversal
4    # in-graph modification (no return values)
5    if hasCycle(nodes,start):
6      print('cycle detected')
7      return
8
9    explored = set()
10   frontierS = []
11   frontierS.append(start)
12   while len(frontierS) > 0:
13     node = frontierS[-1]
14
15     if node in explored:
16       frontierS.pop()
17       continue
18     if len(node.ngbs) == 0 or allIn(node.ngbs, explored):
19       # call the state-less callback here
20       cb(nodes,node)
```

```
21          explored.add(node)
22          frontierS.pop()
23
24        # visit neighbors
25        for ngb in node.ngbs:
26          if ngb not in explored:
27            frontierS.append(ngb)
28
29      return
```

Listing 5.1 shows the routine that implements post-order traversal on a graph, with the last input argument being a generic callback function, to be called upon each node during traversal. To obtain the guided-processing solution, the callback routine computes the value function for the current node based on value functions of its downstream neighbors, which is guaranteed to exist by construction (see Eq. (5.1)). The routine to check for cycles in a graph is also given in Listing 5.2.

Listing 5.2: In-graph cycle detection using depth-first search (DFS) in Python

```
1  def hasCycle(nodes,start):
2    chain = []
3
4    explored = set()
5    frontierS = []
6    frontierS.append(start)
7    while len(frontierS) > 0:
8      node = frontierS.pop()
9
10     if node in explored:
11     continue
12
13     # maintaining the chain invariance
14     while len(chain)>0 and node not in chain[-1].ngbs:
15       chain.pop()
16     # new node bites the chain!
17     if anyIn(node.ngbs,chain):
```

```
18          return True
19        chain.append(node)
20
21        explored.add(node)
22
23        # visit neighbors
24        for ngb in node.ngbs:
25          if ngb not in explored:
26            frontierS.append(ngb)
27
28      return False
```

At each node/iteration, with all downstream neighbors processed as the result of the post-order traversal, the guided-processing equations are given as follows.

$$\begin{aligned} V_i(\pi_i) &= \min(C_M\pi_i, C_A(1-\pi_i)), \text{ if } \mathcal{N}(i) = \emptyset \\ V_i(\pi_i) &= \min(C_M\pi_i, \{\lambda(D_n - d_n) + \mathbb{E}[V_n(\pi_n(Y_n, \pi_i))] : n \in \mathcal{N}(i)\}), \text{ else} \\ V_0(\pi_0) &= \lambda \sum_i d_i + \lambda(D_1 - d_1) + \mathbb{E}[V_1(\pi_1(Y_1, \pi_0))] \end{aligned} \tag{5.1}$$

where it is assumed that the miss and false-alarm cost of all nodes are the same and, again, denoted by C_M, C_A. Similarly, D_n and d_n are the resource on-cost (for extracting feature Y_n) and off-cost of node n. The symbols i, V_i, π_i denote the current node, its value function, and its posterior probability, respectively. $\mathcal{N}(i)$ denotes the set of all (downstream) neighbors of node i. Nodes with no neighbor are last/terminal nodes.

The corresponding optimal decision functions are given as follows.

$$\begin{aligned} \delta_i^*(\pi_i) &= \begin{cases} 0, \text{ if } V_i(\pi_i) = C_M\pi_i \\ 1, \text{ else} \end{cases}, \text{ if } \mathcal{N}(i) = \emptyset \\ \delta_i^*(\pi_i) &= \begin{cases} F_n, \text{ if } V_i(\pi_i) = \lambda(D_n - d_n) + \mathbb{E}[V_n(\pi_n(Y_n, \pi_i))], n \in \mathcal{N}(i) \\ 0, \text{ else} \end{cases}, \text{ else} \end{aligned} \tag{5.2}$$

where F_n is the decision to choose a (downstream) neighbor n for subsequent feature extraction. For practical implementation, the solution in (5.2) is

rewritten in terms of the generalized activation probability q_i as follows.

$$\begin{aligned} q_i(\pi_{i-1}) &\triangleq \{\mathrm{P}(\delta_i^* = a|\pi_{i-1}) : a = 0, 1\}, \text{ if } \mathcal{N}(i) = \emptyset \\ q_i(\pi_{i-1}) &\triangleq \{\mathrm{P}(\delta_i^* = a|\pi_{i-1}) : a = 0, F_n, n \in \mathcal{N}(i)\}, \text{ else} \end{aligned} \tag{5.3}$$

where the notation π_{i-1} is slightly abused to denote the *union* of all admissible priors of node i's parents, i.e.

$$\begin{aligned} \pi_{i-1} &\triangleq \cup_{p\in\mathcal{P}(i)}\pi_p \\ &= [\min_{p\in\mathcal{P}(i)} \pi_{Lp}, \max_{p\in\mathcal{P}(i)} \pi_{Up}] \end{aligned} \tag{5.4}$$

where $\mathcal{P}(i)$ denotes the set of node i's parents and π_{Lp}, π_{Up} are the lower and upper bounds on the admissible prior at a parent node p. The union operation follows from the assumption that triggers from parent nodes are mutually exclusive.

5.3 System simulation

To illustrate the algorithm introduced in Section 5.2, a graph-based system is simulated as shown in Fig. 5.1. Each node in the graph is a detection module, and its value is the resource on-cost D_i. It is herein assumed that the on-costs are unique and hence can be conveniently used to identify a node/detector. All off-costs $d_i, i = 1, \ldots, 9$ are assumed to be 1% of the corresponding on-cost. The graph is organized into four layers, with the on-costs of nodes in the next layer being approximately 10-fold of those in the current layer, in rough agreement with the physical system described in Chapter 4. Finally, an arrowed edge between nodes represents the parent-child relationship in which only a parent node can trigger its child (i.e. downstream neighbor) nodes. For instance, node 10 can choose between nodes $100, 105, 110,$ and 130 for subsequent processing.

The likelihood models (observation distributions under target absence and presence, i.e. P_0 and P_1, respectively) for all nodes/detectors are assumed to be Gaussian with different means and variances, as shown in Fig. 5.2. Nodes in later layers/with higher on-costs have stronger discrimination power. Unlike Chapter 4, uncertainties in these *synthetic* models are herein not considered, but can be easily incorporated if desired. Finally, similar to the setup

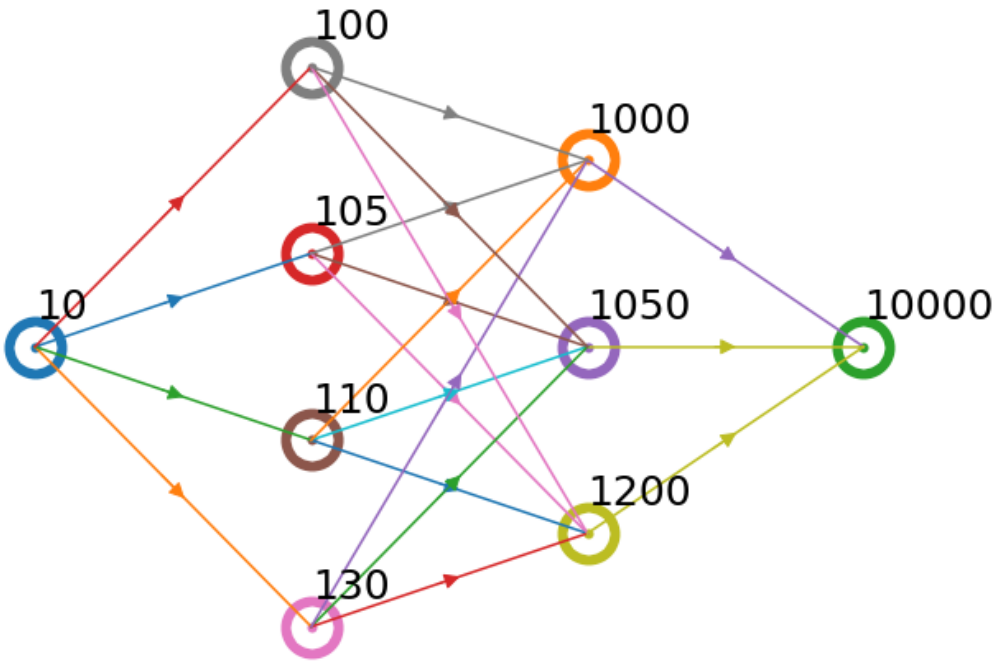


Figure 5.1: A hypothetical graph-based, guided-processing system. Each node is a detection module labeled with its resource on-cost (off-cost is assumed to be 1% of the on-cost, for simplicity), e.g. node 10 costs 10 resource units to execute. The corresponding likelihood model for each node is given in Fig. 5.2.

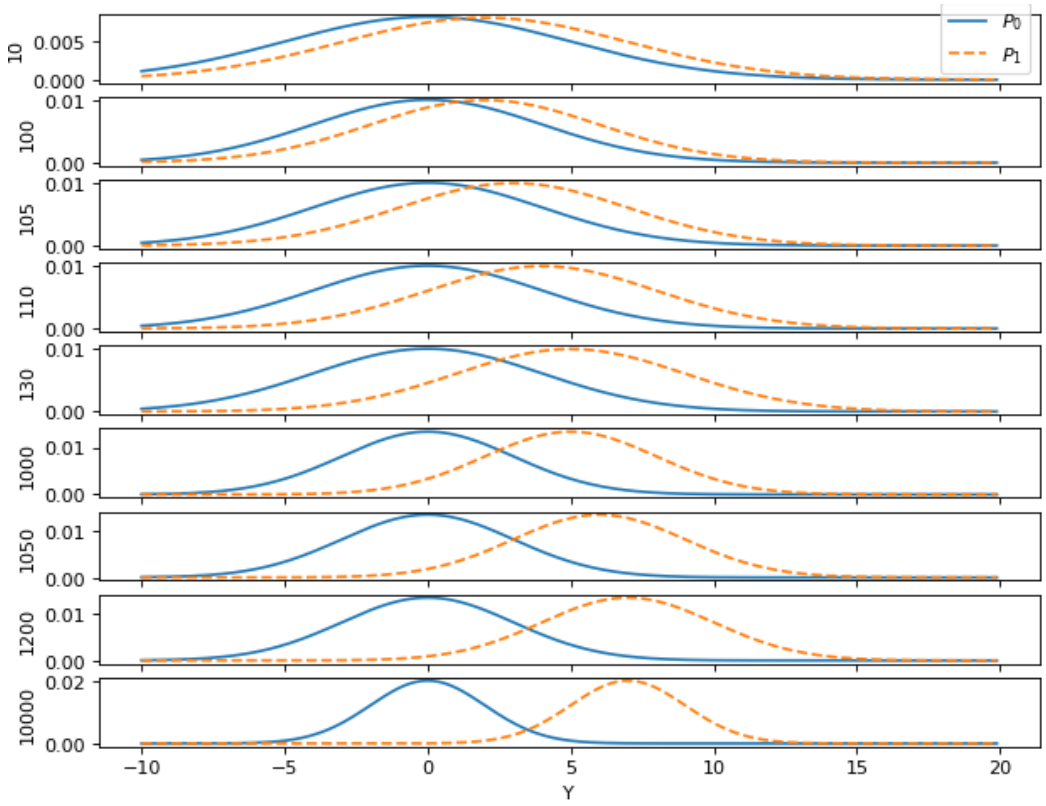


Figure 5.2: Likelihood models, with better discriminative power on more expensive modules.

in Section 4.4, it is assumed that $C_M = 3, C_A = 1$ to emphasize that the miss risk is higher in the rare-event setting.

Figure 5.3 shows the resulting system risk (a total of Bayes risk and weighted resource consumption) of the system using either the guided-processing, or the duty-cycling approach with duty-cycle $\rho = 1$ and $\rho = 0$. Higher system risk corresponds to lower energy-efficiency. It is evident that, except for very high π_0, the guided-processing approach is more energy-efficient than ideal bounds on the duty-cycling one. Figures 5.4, 5.5, and 5.6 show further comparisons in term of resource consumption, miss rate, and false-alarm rate, respectively.

The optimal policies for all nodes/detectors can be computed using (5.2), and the corresponding activation probabilities needed for the adaptive implementation are given in Fig. 5.7. For a given Module i, its activation probability q_i is represented by multiple functions over the prior π_{i-1}. The functions sum to one and are labeled according to the decisions available at that module. Each function maps the prior to the probability of the corre-

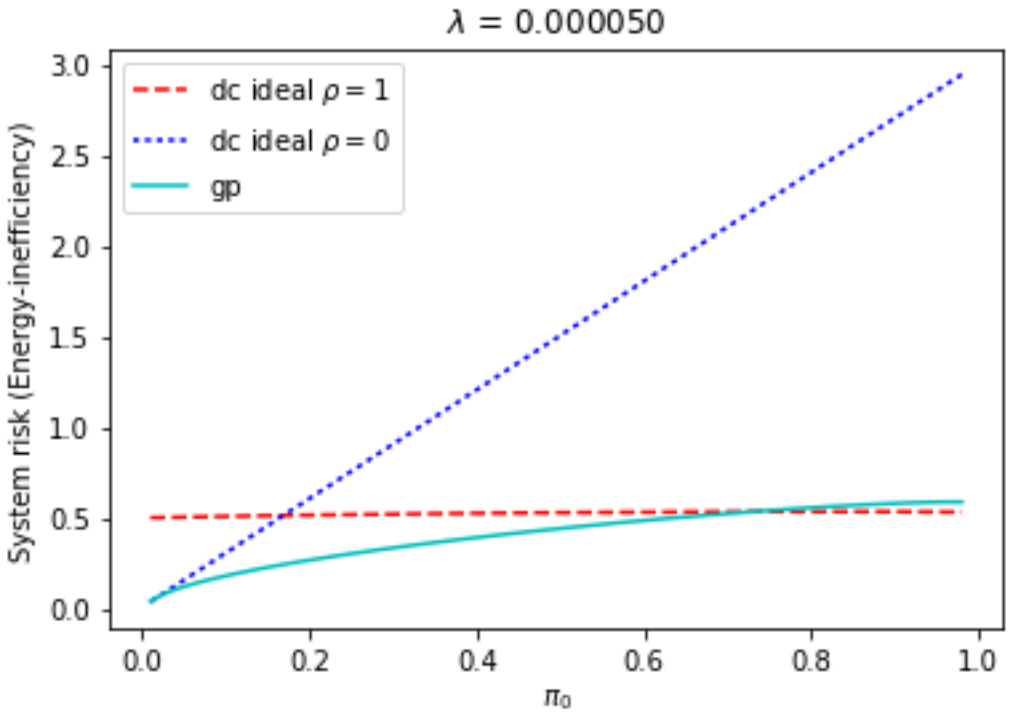


Figure 5.3: Comparison of system risks across all prior π_0 between guided-processing (gp) and duty-cycling (dc). The energy-efficiency of the guided-processing approach *decreases* as $\pi_0 \to 1$.

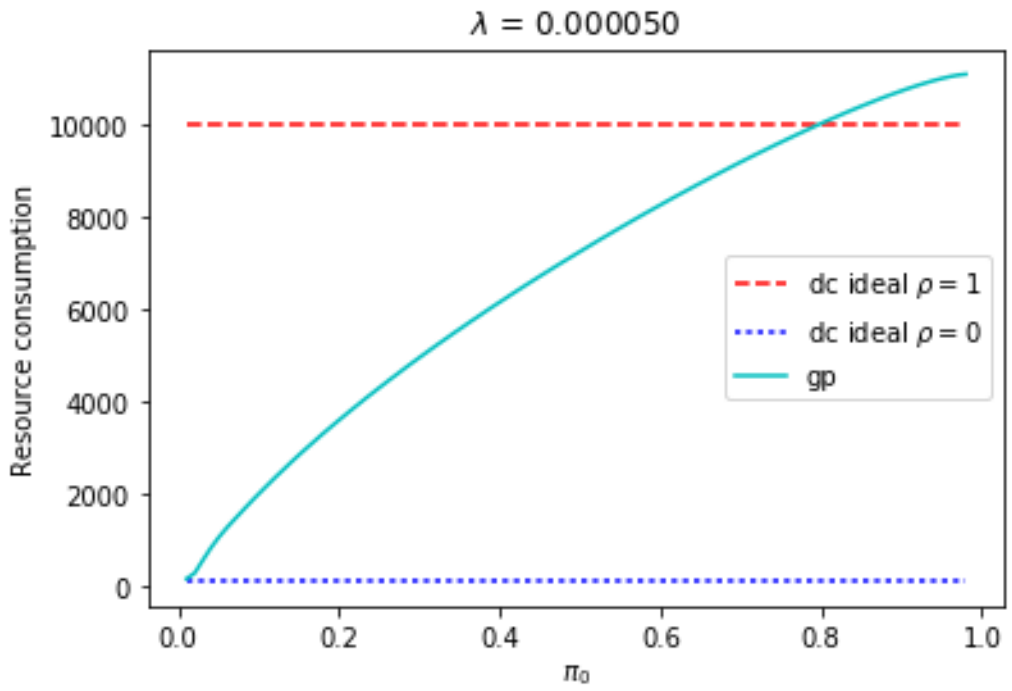


Figure 5.4: Comparison of resource consumption across all prior π_0 between guided-processing (gp) and duty-cycling (dc).

sponding decision being selected. For instance, at Module 10, there are early negative decision (labeled by 0) and the decisions to select Modules 100, 110, or 130 as the next downstream module. Note that π_{10-1} denotes the prior at Module 10. It is worth noting that there is a common pattern in the first five activation probabilities, which provides an intuitive interpretation for the optimal policies. Namely, a node is more likely to choose its resource-expensive, but high-performance, child node if provided with an uncertain (i.e. $\pi_i \in 0.5 \pm \epsilon$ for some $\epsilon > 0$) prior (posterior from its parent nodes), and to only consider cheaper alternatives otherwise.

5.4 Summary and conclusions

This chapter shows that the resource management problem on graph-based systems can also be solved by applying the guided-processing principle. The solution is achieved by generalizing the backward recursion developed for cascade systems to one that employs a post-order traversal. Future work will apply this generalization to optimize even more resource-intense systems, e.g.

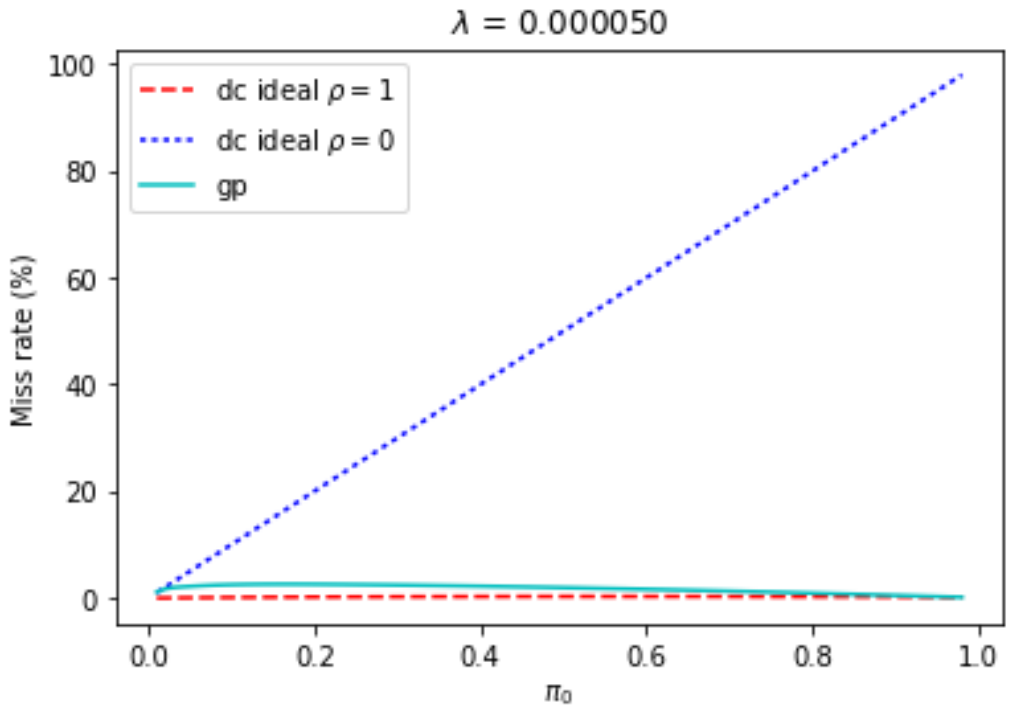


Figure 5.5: Comparison of miss rate across all prior π_0 between guided-processing (gp) and duty-cycling (dc). Note that gp maintains very close miss rate to dc with $\rho = 1$.

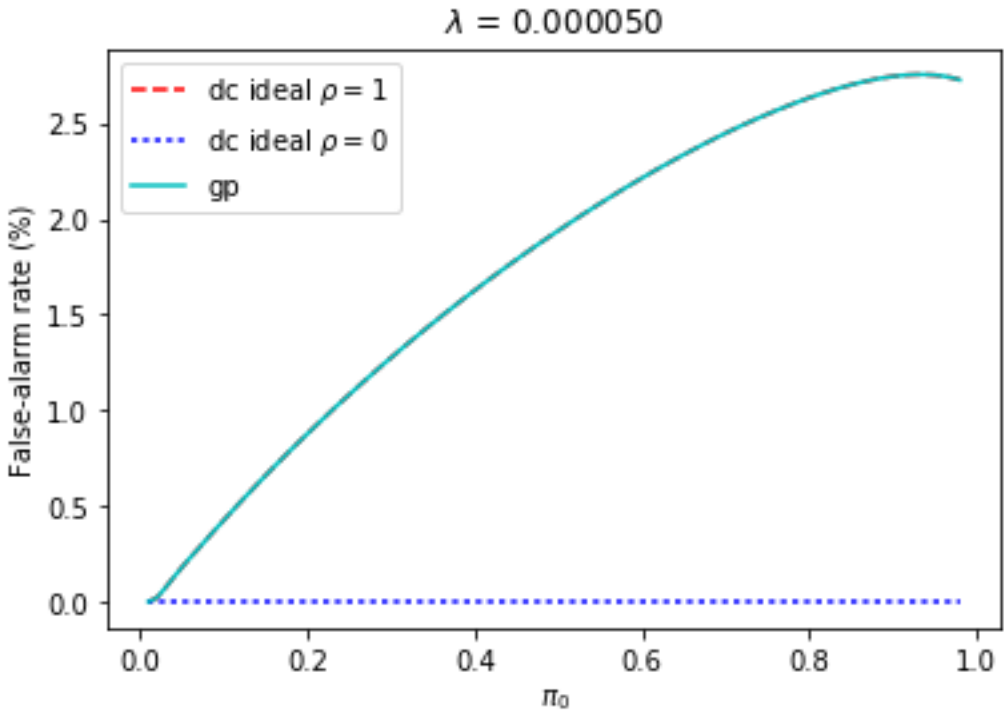


Figure 5.6: Comparison of false-alarm rate across all prior π_0 between guided-processing (gp) and duty-cycling (dc). Note that gp has the same false-alarm rate as dc with $\rho = 1$.

a video-based object recognizer, where a graph-based architecture is likely needed.

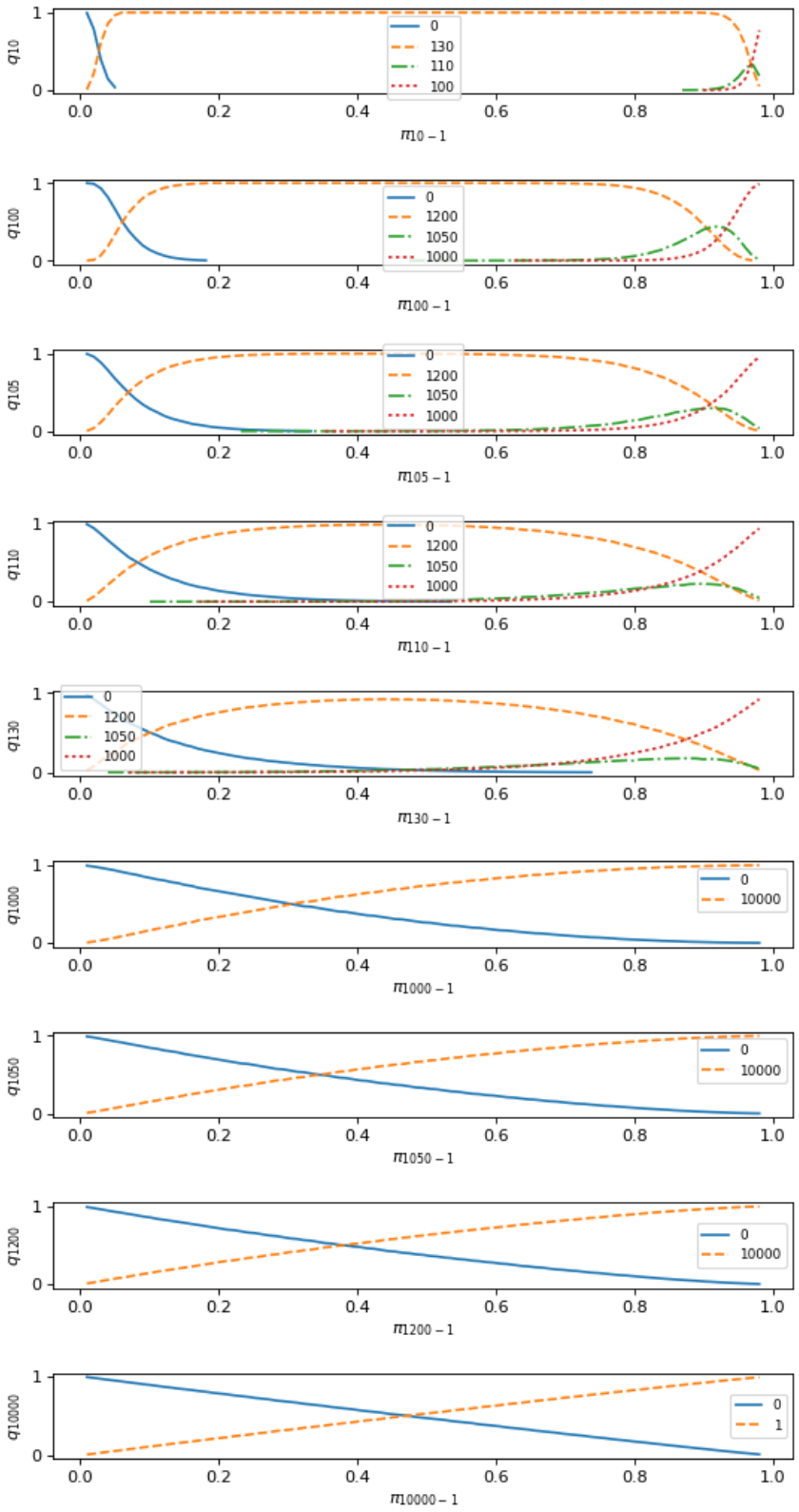


Figure 5.7: Activation probabilities of all modules. The activation probability q_i of module i is represented by multiple functions, which sum to one, over the prior π_{i-1}. Each function maps the prior to the probability of the corresponding decision being selected.

CHAPTER 6

THE FEATURE-SHARING PRINCIPLE

6.1 Overview

Traditional distributed detection systems are often designed for a single application. However, with the emergence of the Internet of Things (IoT) paradigm, next-generation systems are expected to be a shared infrastructure for multiple applications, and hence require rethinking of the overall system design.

To support multiple tasks seamlessly, a detection system needs to be modularly designed, i.e. made up of components that are reusable for various applications with different objectives and constraints. A similar view is shared by the TerraSwarm Research Center [1], whose aim is to create software components that serve as building blocks for IoT application developers. Likewise, Atzori et al. [20] proposed a service-oriented architecture for the IoT, where an ecosystem of services lays the foundation for IoT applications to be built on top.

Like many system design problems, resource-efficiency is an important objective in the design of multi-task detection systems [69]. A well-known resource-efficient, modular design is the cascade structure, which consists of a collection of detection modules in tandem. The design has been applied successfully in the single-application context, e.g. face detection [67]. However, we propose that the cascade also has a great potential in the multiple-application context. Namely, thanks to its modular design, modules in the cascade could be shared between applications. In addition, the output of a

Copyright 2017 IEEE. Some of the material in this chapter has been reproduced, with permission, from: Long N. Le, Douglas L. Jones. "Feature-Sharing in Cascade Detection Systems with Multiple Applications." IEEE Journal of Selected Topics in Signal Processing, Special Issue on Cooperative Signal Processing for Heterogeneous and Multi-Task Wireless Sensor Networks, 2017 [96].

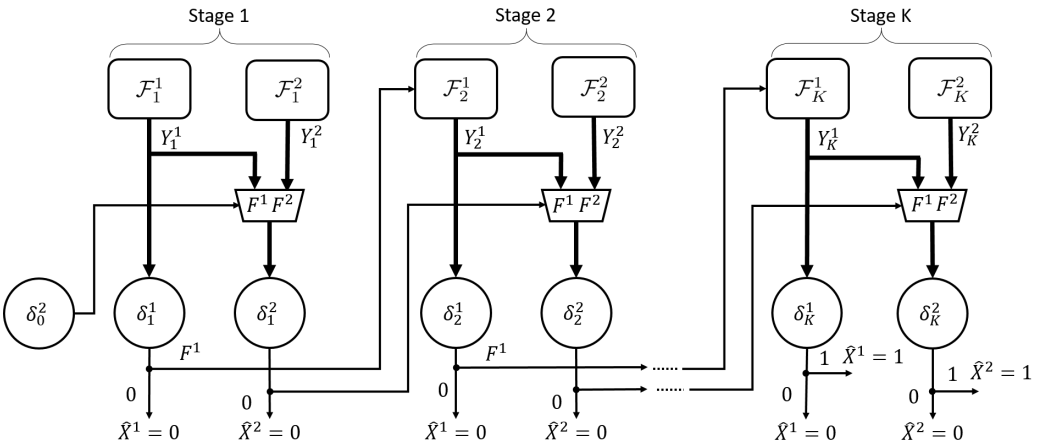


Figure 6.1: The cascade detection system with 2 applications (indexed by superscripts) and K stages/layers (indexed by subscripts). For stage i of application j, $\mathcal{F}_i^j$ denotes the feature extractor and δ_i^j denotes the binary decision of a detector. The feature itself is denoted by Y_i^j. X^j is the (detection) target state, and $\hat{X}^j$ is the prediction about X^j by a detector.

module can be used to dynamically guide/control the execution of others in the system, providing necessary degrees of freedom to optimize the trade-off between system resource consumption and detection performance.

In this chapter, we specifically study the multi-application cascade detection system whose model is shown in Fig. 6.1. It is assumed that there are two applications and K layers in the system. The system is illustrated in Fig. 6.1, where the superscripts are used to index applications, and the subscripts are used to index stages.

Each layer of the cascade is occupied by detection modules/detectors from both applications. Ignoring the application index (superscript) for now, a detector at stage i consists of a feature extractor $\mathcal{F}_i$, which produces the feature Y_i, and a decision rule δ_i, which takes Y_i and all previous features $Y_1, \ldots, Y_{i-1}$ as input. δ_i outputs different values depending on both the application and the stage (see Eq. (6.1) and (6.2)). X is the state of the (detection) target, which takes value 1 when the target is present, and 0 otherwise. Finally, $\hat{X}$ denotes the prediction of X by the detector.

Of the two applications, we let one be the *primary* and the other be the *secondary*, denoted by superscript indices 1 and 2 in Fig. 6.1, respectively. The primary application is the one whose feature extractors produce *universal features* that are sharable. In contrast, features produced by the sec-

ondary application are not universal and hence assumed to produce no value in sharing. This distinction is illustrated in Fig. 6.1, where primary features $Y_i^1, i = 1, \ldots, K$ can be shared with the secondary application, but secondary features $Y_i^2, i = 1, \ldots, K$ can only be used by the secondary application. Examples of universal features for audio applications are signal energy, the Mel-frequency cepstrum coefficient (MFCC) [97] and time-varying sinusoidal features [29], which have been used extensively in various audio/speech inference applications. Examples of secondary features are internal representation in neural networks, such as that of autoencoders. It is worth noting that the definition of primary and secondary applications here has no relevance to the practical importance of each, and the two-application assumption is meant for simplicity, without loss of generality. In fact, our result can be easily generalized to an arbitrary number of applications in each class (primary, secondary).

The decisions of the primary application δ_i^1 can take on the following values depending on the layer/stage.

$$\begin{aligned} \delta_i^1 &= \begin{cases} 0: \text{ stop and declare } \hat{X}^1 = 0 \text{ (negative)} \\ F^1: \text{ use the primary feature next} \end{cases} \\ & i = 1, \ldots, K-1 \\ \delta_K^1 &= \begin{cases} 0: \text{ declare } \hat{X}^1 = 0 \text{ (negative)} \\ 1: \text{ declare } \hat{X}^1 = 1 \text{ (positive)} \end{cases} \end{aligned} \tag{6.1}$$

Note that only negative decisions, i.e. $\hat{X} = 0$, are allowed at intermediate stages ($i = 1, \ldots, K-1$) since the goal is not to make the final decision (which is reserved for the last stage with the best performance) but to screen out early negative instances, which is more likely in a rare-target setting.

Since the secondary application has access to both primary and secondary features, its decisions δ_i^2 at intermediate stages have more options and are

given as follows.

$$\delta_0^2 = \begin{cases} F^1: \text{ use the primary feature next} \\ F^2: \text{ use the secondary feature next} \end{cases}$$

$$\delta_i^2 = \begin{cases} 0: \text{ stop and declare } \hat{X}^2 = 0 \text{ (negative)} \\ F^1: \text{ use the primary feature next} \\ F^2: \text{ use the secondary feature next} \end{cases} \qquad (6.2)$$

$$i = 1, \ldots, K-1$$

$$\delta_K^2 = \begin{cases} 0: \text{ declare } \hat{X}^2 = 0 \text{ (negative)} \\ 1: \text{ declare } \hat{X}^2 = 1 \text{ (positive)} \end{cases}$$

Namely, intermediate decisions include feature selection and (early) negative decision making. Note that δ_0^2 is the decision that occurs before any features are observed, and thus is restricted from making early negative decisions.

The cascade structure has been studied before in the literature. For instance, the seminal work by Viola and Jones [67] showed empirically that such a design is very effective in detecting rare targets in a large dataset, and was also proposed as a resource-efficient approach for stream mining by Turaga et al. in [68]. However, existing works either 1) offer solutions that have limitations or 2) focus on a single application. Our study here involves the cascade structure with multiple applications, investigates the potential of sharing features between them, and offers a solution that does not have limitations of prior works. For instance, our resource-consumption model is more accurate than existing works, which often suffer from a 'nebulous' resource-consumption model that is inapplicable in practice. Furthermore, it is observed that there are inherent uncertainties in some features of the cascade, and an approach is proposed to address them. Besides optimizing parameters of the cascade, we also show that, under mild conditions, the cascade design itself is optimal. That is, adding additional degrees of freedom such as early positive decisions to the cascade structure does not improve its performance.

The rest of the chapter is organized as follows. Section 6.2.1 discusses feature models and their potential uncertainties, then Section 6.2.2 presents our formulation and solution for the multi-application cascade system. A system

simulation and final remarks are given in Sections 6.3 and 6.4, respectively.

6.2 Optimizing the multiple-application cascade system

6.2.1 Feature models

The discussion on feature models for the single-application case translates nicely here, and the reader is redirected to Section 4.3.1 for details.

6.2.2 System model and optimization

Optimizing the cascade system amounts to finding optimal decision rules $\delta_{1:K}^{1:2}$ that jointly minimize the proposed system's Bayes risk R_B of incorrect decisions subject to an expected system resource (e.g. energy) constraint e.

$$\begin{aligned} &\min_{\delta_{1:K}^{1:2}} R_B(\delta_{1:K}^{1:2}) \triangleq \sum_{j=1}^{2} R_B^j(\delta_{1:K}^j) \\ &s.t.\ E(\delta_{1:K}^{1:2}) \triangleq \sum_{j=1}^{2} E^j(\delta_{1:K}^j) \leq e \end{aligned} \tag{6.3}$$

where E is the expected system resource consumption. It is assumed that the total Bayes risk and expected resource consumption of the system is the sum from those of the individual application, i.e. R_B^j and E^j, $j = 1, 2$. The Lagrangian technique can be used to convert the constrained optimization problem (6.3) into the following unconstrained, but regularized, one:

$$\begin{aligned} \min_{\delta_{1:K}^{1:2}} R(\delta_{1:K}^{1:2}) &\triangleq \sum_{j=1}^{2} R^j(\delta_{1:K}^j) \\ &\triangleq \sum_{j=1}^{2} (\lambda E^j + R_{K,\mathrm{A}}^j + \sum_{i=1}^{K} R_{i,\mathrm{M}}^j) \end{aligned} \tag{6.4}$$

where the parameter λ, which depends on the resource constraint e, couples the resource consumptions of all stages together and R denotes the *system risk* (with R^j denotes the risk of application j), which is a mea-

sure of the combined detection performance and resource consumption. The Bayes risk R_B^j has been broken down into multiple terms. For application j, $R_{i,M}^j, i = 1, \ldots, K-1$ are the miss (false negative) risks due to early negative decisions at intermediate stages. $R_{K,M}^j, R_{K,A}^j$ are the miss and false-alarm (false positive) risks due to incorrect decisions at the last stage. Note that the system has no false-alarm risk at intermediate stages, since the cascade structure does not allow early positive decisions to be made. Such a constraint is useful to reduce the false-positive decision rate especially under model uncertainty and when the target is rare.

For application j, the expected resource consumption at stage i is the resource cost of feature extraction, denoted by D_i^j, weighted by the probability of the feature being selected by the previous stage. Hence,

$$\begin{aligned} E^1 &\triangleq D_1^1 + \sum_{i=1}^{K-1} D_{i+1}^1 \mathrm{P}(\delta_i^1 = F^1) \\ E^2 &\triangleq \sum_{i=0}^{K-1} D_{i+1}^2 \mathrm{P}(\delta_i^2 = F^2) \end{aligned} \tag{6.5}$$

where D_1^1 is weighted by 1 because the first-stage primary feature is always extracted. Note that D_i^j can be measured in practice by profiling the feature-extraction process.

The solution to Problem (6.4) is given by the following theorem.

Theorem 6.1. *(The optimal decision rules for all applications in the cascade) For the primary application,*

$$\begin{aligned} \delta_i^{1*}(\pi_i^1) &= \begin{cases} 0, \pi_i^1(y_{1:i}^1) < \tau_i^{1*} \\ F^1, \ \textit{else} \end{cases} \\ i &= 1, \ldots, K-1 \\ \delta_K^{1*}(\pi_K^1) &= \begin{cases} 0, \pi_K^1(y_{1:K}^1) < \tau_K^{1*} \\ 1, \ \textit{else} \end{cases} \end{aligned} \tag{6.6}$$

For the secondary application,

$$
\begin{aligned}
\delta_0^{2*}(\pi_0^2;\pi_0^1) &= F^1, \forall \pi_0^2, \forall \pi_0^1 \\
\delta_i^{2*}(\pi_i^2;\pi_i^1) &= \begin{cases} 0, \pi_i^2(y_{1:i}^2) < \tau_i^{2*}, \pi_i^1 < \tau_i^{1*} \\ F^2, \pi_i^2(y_{1:i}^2) \geq \tau_i^{2*}, \pi_i^1 < \tau_i^{1*} \\ 0, \pi_i^2(y_{1:i}^2) < \eta_i^{2*}, \pi_i^1 \geq \tau_i^{1*} \\ F^1, \pi_i^2(y_{1:i}^2) \geq \eta_i^{2*}, \pi_i^1 \geq \tau_i^{1*} \end{cases} \\
& i = 1, \ldots, K-1 \\
\delta_K^{2*}(\pi_K^2) &= \begin{cases} 0, \pi_K^2(y_{1:K}^2) < \tau_K^{2*} \\ 1, \ \textit{else} \end{cases}
\end{aligned}
\tag{6.7}
$$

where $\tau_i^{j*}, \eta_i^{j*} \in [\pi_{Li}^j, \pi_{Ui}^j]$ *are the optimal thresholds at stage* i *of application* j*, provided that*

$$
C_M^2 \left\{ \mathbb{E}[\pi_i^2(Y_i^1)] - \mathbb{E}[\pi_i^2(Y_i^2)] \right\} \leq \lambda D_i^2, i = 1, \ldots, K \tag{6.8}
$$

with C_M^2 *defined in Corollary 6.1.*

Proof. See Appendix B.1. □

Equation (6.6) in Theorem 6.1 shows that, for the primary application, the posterior probabilities of intermediate stages can be used to guide the execution of subsequent stages by thresholding them to decide whether to stop or extract more primary features in the next stage. The final stage has a standard detection rule, with the posterior probability being thresholded to make a prediction about the target state.

Furthermore, Eq. (6.7) shows that the decision rules at intermediate stages of the secondary application are not only a function of π_i^2, but are also parameterized[1] by π_i^1. If $\pi_i^1 \geq \tau_i^{1*}$, then according to (6.6), the next-stage primary feature is available, and the optimal decision always selects this feature over the secondary feature (feature sharing) for the next stage (as long as π_i^2 is above the threshold for early negative decision η_i^{2*}). Since the first-stage primary feature is always available, it is always selected by δ_0^{2*}. Otherwise, if the primary feature is not available because $\pi_i^1 < \tau_i^{1*}$, then the secondary application falls back to selecting the secondary feature, assuming

[1]The term after the semicolon

π_i^2 is above the threshold for early negative decision τ_i^{2*}. Note that the thresholds for early negative decisions are different under each case. Finally, the final stage's decision is simply a standard detection rule.

The structure of (6.7) favors feature-sharing whenever possible. This policy is optimal provided that additional constraints in (6.8) on the parameters of the cascade hold. Intuitively, (6.8) requires that the expected increase in the secondary miss risk due to the shared/primary feature usage is relatively small compared to the resource cost of extracting the secondary feature.

Finally, the optimal threshold values $\{\tau_i^{j*}, j = 1, 2\}$, which are critical in this trade-off between performance and resource cost, can be found using Algorithm 2 in Appendix B.3.

Given the above optimal decisions, Corollary 6.1 quantifies the optimal performance of the multi-application cascade system.

Corollary 6.1. *(Optimal performance of applications in the cascade) For the primary application,*

$$R^{1*}(\pi_0^1) \triangleq R^1(\delta_{1:K}^{1*}, \pi_0^1) = V_0^1(\pi_0^1) \tag{6.9}$$

where $V_0^1(\pi_0^1)$ is the result of the following recursion

$$\begin{aligned}
V_K^1(\pi_K^1) &\triangleq \min(\underbrace{C_M^1 \pi_K^1}_{\textit{miss risk}}, \underbrace{C_A^1(1-\pi_K^1)}_{\textit{false-alarm risk}}), \pi_K^1 \in [0,1] \\
V_i^1(\pi_i^1) &\triangleq \min(C_M^1 \pi_i^1, \underbrace{\lambda D_{i+1}^1 + \mathbb{E}[V_{i+1}^1(\pi_{i+1}^1(Y_{i+1}^1, \pi_i^1))]}_{\textit{expected next-stage primary value function}}), \\
&\pi_i^1 \in [\pi_{Li}^1, \pi_{Ui}^1], i = 1, \ldots, K-1 \\
V_0^1(\pi_0^1) &\triangleq \lambda D_1^1 + \mathbb{E}[V_1^1(\pi_1^1(Y_1^1, \pi_0^1))]
\end{aligned} \tag{6.10}$$

And the corresponding optimal thresholds are given by

$$\begin{aligned}
\tau_K^{1*} &= C_A^1/(C_A^1 + C_M^1) \\
\tau_i^{1*} &= \max(\pi_{Li}^1, \min(\pi_{Ui}^1, \min\{\pi_i^1 : V_i^1(\pi_i^1) - C_M^1 \pi_i^1 < 0\})), \\
&\quad i = 1, \ldots, K-1
\end{aligned} \tag{6.11}$$

For the secondary application,

$$R^{2*}(\pi_0^2; \pi_0^1) \triangleq R^2(\delta_{1:K}^{2*}, \pi_0^2; \pi_0^1) = V_0^2(\pi_0^2; \pi_0^1) \tag{6.12}$$

where $V_0^2(\pi_0^2;\pi_0^1)$ *is the result of the following recursion*

$$
\begin{aligned}
V_K^2(\pi_K^2) &\triangleq \min(C_M^2\pi_K^2, C_A^2(1-\pi_K^2)), \pi_K^2 \in [0,1] \\
V_i^2(\pi_i^2;\pi_i^1) &\triangleq \begin{cases} \min(C_M^2\pi_i^2, \\ \quad \underbrace{\lambda D_{i+1}^2 + \mathbb{E}[V_{i+1}^2(\pi_{i+1}^2(Y_{i+1}^2,\pi_i^2);\pi_i^1)]}_{\substack{\textit{expected next-stage secondary value function} \\ \textit{using the secondary feature}}}), \\ \quad \textit{if } \pi_i^1 < \tau_i^{1*} \\ \min(C_M^2\pi_i^2, \\ \quad \underbrace{\mathbb{E}[V_{i+1}^2(\pi_{i+1}^2(Y_{i+1}^1,\pi_i^2);\pi_i^1)]}_{\substack{\textit{expected next-stage secondary value function} \\ \textit{using the shared primary feature}}}), \\ \quad \textit{else} \end{cases} \\
\pi_i^2 &\in [\pi_{Li}^2, \pi_{Ui}^2], i = 1,\ldots,K-1 \\
V_0^2(\pi_0^2;\pi_0^1) &\triangleq \mathbb{E}[V_1^2(\pi_1^2(Y_1^1,\pi_0^2);\pi_0^1))]
\end{aligned}
\tag{6.13}
$$

and the corresponding optimal thresholds are given by

$$
\begin{aligned}
\tau_K^{2*} =& C_A^2/(C_A^2+C_M^2) \\
\tau_i^{2*} =& \max(\pi_{Li}^2, \min(\pi_{Ui}^2, \\
& \qquad \min\{\pi_i^2 : V_i^2(\pi_i^2;\pi_i^1) - C_M^2\pi_i^2 < 0\})), \\
& \textit{if } \pi_i^1 < \tau_i^{1*} \\
\eta_i^{2*} =& \max(\pi_{Li}^2, \min(\pi_{Ui}^2, \\
& \qquad \min\{\pi_i^2 : V_i^2(\pi_i^2;\pi_i^1) - C_M^2\pi_i^2 < 0\})), \\
& \textit{else} , \\
& i = 1,\ldots,K-1,
\end{aligned}
\tag{6.14}
$$

where $C_M^j, C_A^j, j = 1,2$ *are the costs associated with miss and false-alarm decisions of application* j*.*

Corollary 6.1 shows that the optimal performance achieved by each application can be found using a recursive procedure. The procedure has K iterations, each corresponding to a stage in the system. Starting from the last stage K and proceeding backward to 0, the value function V_i^j is recursively updated (see (6.10) and (6.13)). The last-stage value function V_K^j is the minimum of the miss and false-alarm risks across π_K. An intermediate-stage

value function $V_i^j, i = 1, \ldots, K-1$ is the minimum of the miss risk and the *expected next-stage* value function, which requires the probabilistic updates in (4.4),(4.5). The final value function V_0^j is the minimal risk achievable by an application.

Note that the secondary application's value function at an intermediate stage is given by different expressions depending on the availability of the primary feature for the next stage, i.e. $\pi_i^1 \lessgtr \tau_i^{1*}$ (see (6.13)). If the primary feature is not available for the next stage, then the expected next-stage value function is taken with respect to the secondary feature, whose extraction cost (λD_{i+1}^2) is also included. Otherwise, the expected next-stage value function does not contain the resource cost to extract the secondary feature and the expectation is taken with respect to the primary feature.

Once a value function is known, then the corresponding optimal threshold can be found using just arithmetic operations, i.e. comparing the value function with the miss risk (see (6.11) and (6.14)). For the last stage K, the optimal threshold can be given in closed form. Note that the intermediate stages' thresholds are capped between the upper and lower bounds due to model uncertainty (see Section 6.2.1). For the secondary application, depending on the availability of the primary feature, the thresholds for early negative decisions are different, and hence denoted differently (τ_i^2 and η_i^2, respectively). Intuitively, if the primary threshold is low, i.e. the primary application consumes most of the resource budget, then the secondary application is more inclined to use the shared primary feature due to low resource budget. On the other hand, if the primary threshold is high, i.e. the secondary application has most of the resource budget, then the secondary feature is used more to reduce its miss and false-alarm risks.

The discussion so far has focused on optimizing parameters of the cascade design. A natural next question to ask is whether the constraints of the cascade design can be relaxed to further improve performance. Namely, would introducing additional degrees of freedom, i.e. early positive decisions in intermediate stages, to the cascade *always* improve its performance? Intuitively, when model uncertainties of intermediate stages are accounted for (see Section 6.2.1), and it is known *a priori* that the target is rare, early positive decisions are likely to have higher risk and hence are discouraged. Therefore, introducing additional early positive decisions does *not* always improve the performance of the cascade. The precise conditions for which

the cascade design itself is optimal are given by the following proposition.

Proposition 6.1. *(Optimality of the cascade design) With model uncertainty, introducing additional early positive decisions in intermediate stages of the cascade does not improve performance, as long as*

$$\underbrace{\max\{\pi_i^j : V_i^j(\pi_i^j) - C_A^j(1-\pi_i^j) < 0\}}_{\textit{optimal threshold for early positive decision}} > \pi_{Ui}^j, \quad i = 1, \ldots, K-1, j = 1, 2 \tag{6.15}$$

Proof. See Appendix B.2. □

The left-hand side of (6.15) is the optimal threshold corresponding to an early positive decision. Namely, these additional decisions also have threshold-based optimal policies (see Appendix B.2), and a posterior probability *above* such a threshold shall trigger an early positive decision. If such a threshold is above the upper bound on the posterior probability at a stage, then its early positive decision is never selected, and hence does not affect the performance of the cascade.

6.3 System simulation

This section applies the theory developed in Section 6.2 to design a multi-application, acoustic detection system.

6.3.1 Hardware components

The hardware components needed for an acoustic sensing system are listed in Table 6.1, along with their power consumption (from the datasheets). Note that these are commercial-off-the-shelf (COTS) components, without any customization. The sensor's brain (supplied at 3.6 V) is Silicon Labs's WGM110, which is a low-power wireless (wifi) chip that includes a low-power 12-bit ADC, an ARM Cortex-M3 processor, and a wifi module (among others). All the control logic and the (digital) signal processing software are assumed to be implemented on this general-purposed processor, without any

ASIC[2] or DSP.[3] In addition, a microphone and a preamp are also a part of the acoustic sensor. The power consumption of the microphone, the preamp circuit, and the ADC altogether is 3.6 mW and considered as the baseline of the system. Data collected by the sensor are transmitted downstream to the client, which is a ML100G-10 Next Unit of Computing (NUC) from LogicSupply. Power profiling the NUC (using PowerBlade [98]) results in an average power consumption of 4.744 W (at 9 V).

Table 6.1: Power consumption of hardware components of the acoustic sensing system.

Components	**Power consumption (mW)**
Electret Microphone	0.72
Preamp Circuit (OPA344)	1.08
WGM110 12-bit ADC	1.8[a]
WGM110 ARM core	86.4
WGM110 transmission	900
ML100G-10	4744

6.3.2 Software components

6.3.2.1 Primary application (Golden-cheeked Warbler detection)

The detection of the Golden-cheeked Warbler (GCW)'s (type-A) calls [95] is considered as the primary application. Namely, $X^1 = 1$ indicates the presence of a GCW call, and $X^1 = 0$ otherwise. Since the GCW is an endangered bird species, this application has important implications for their conservation.

The application's software is organized into three subtasks: generic energy-based analysis, spectral-based analysis, and temporal-spectral-based analysis. The energy analysis is a low-complexity computation that produces energy-based features useful for detecting acoustic events from silence. The spectral-based analysis takes into account the spectral information about the

[2]application-specific integrated circuit

[3]digital signal processor

[a]From [88], the ADC draws 0.5 mA, which is equivalent to 1.8 mW at 3.6 V.

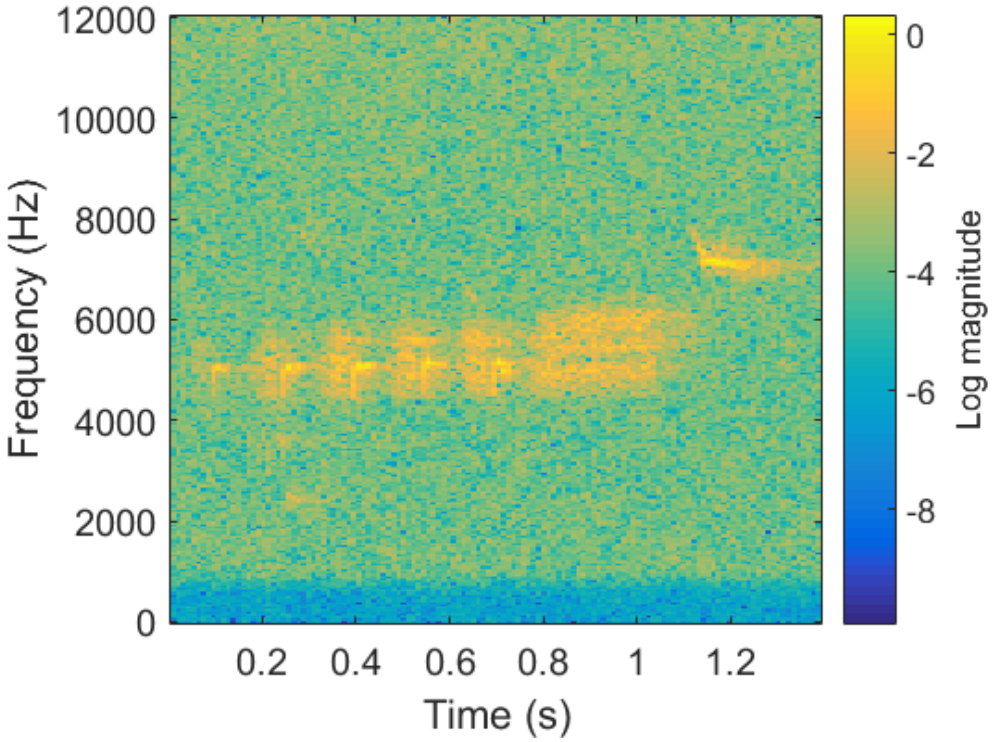


Figure 6.2: Spectrogram of a sample GCW's (type-A) call. Same as Fig. 4.4.

GCW calls, which only has energy in the 4500-6500 Hz and 7000-8000 Hz bands (see Fig. 6.2), to produce band-specific, energy-based features using standard DSP filtering techniques. Finally, the spectral-temporal-based analysis takes into account both the spectral and temporal structure of the GCW call from Fig. 6.2 to produce reliable, indicative features using a template-matching technique. Note that the input into the above analyses is an audio stream (or precisely, its high-dimensional time-frequency representation, see Fig. 6.2), and their output is a scalar score sequence, i.e. a score for each audio frame. Hence, these analyses effectively perform dimensionality reduction.

Since the generic energy analysis has low computational complexity and can help prune out a significant amount of noise-only data from the audio stream early, it is executed on edge/sensor nodes. Only acoustic events are transmitted downstream to clients, where spectral and temporal-spectral-based analyses are further carried out. The system diagram is illustrated in Figure 6.3 and arranged to fit the proposed cascade abstraction. Note that the physical separation (between sensors and clients) does not necessarily

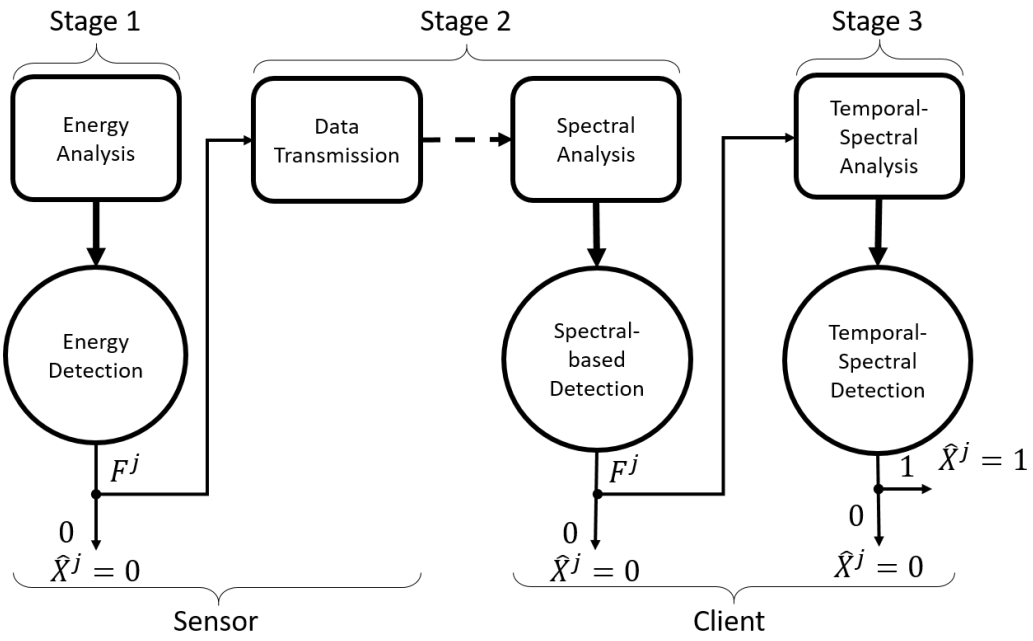


Figure 6.3: The software components of the primary application are organized as a cascade with 3 stages: energy analysis as Stage 1, spectral-based analysis (along with the data transmission) as Stage 2, and temporal-spectral analysis as Stage 3. Note that components of the cascade are distributed across the network, with the dashed line representing a remote connection.

correspond to the logical separation (between stages). For instance, the cost of data transmission on sensors is included into the cost of executing the second stage, along with the cost of spectral-based analysis on clients, since they are both a result of the first-stage decision.

The resource cost parameters at each stage $D_i^1, i = 1, 2, 3$, which can be estimated from values of Table 6.1 and the execution times of the software components, are needed to optimize the resource-performance trade-off. It is assumed that all processing finishes before a periodic deadline, i.e. when buffers (an ADC buffer on the sensor, a task buffer on the client) are full. The average execution time of each task (per audio frame of 32 ms) can be estimated/profiled and is given as follows. The energy analysis takes 16 ms.[4] The average transmission time is 11 ms (500 ms for a 1.5 s event[5]). Finally, the spectral and temporal-spectral analyses take 0.37 μs and 15 ms,

[4]Estimated as half of the frame length.
[5]Profiled on a prototype.

respectively.[6] Hence,

$$\begin{aligned} D_1^1 &= 86.4 \times 0.016 \\ D_2^1 &= 900 \times 0.011 + 4744 \times 0.37 \times 10^{-6} \\ D_3^1 &= 4744 \times 0.015 \end{aligned} \tag{6.16}$$

Our dataset is a 46-minute, 24 kHz audio recording at the field in Rancho Diana, San Antonio's city park. The dataset contains 206 GCW calls (manually identified and labeled), each of whose duration is approximately one second. In addition to GCW calls, the dataset also contains various interferences from other animals' vocalizations, time-varying wind noise, etc., since it is taken directly from field recordings. Precisely, the fraction of GCW calls in the entire dataset is 10.19%. Hence, this detection problem belongs to the rare-target class, where the prior is asymmetrical, i.e. $\pi_0^1 \ll 0.5$. In this simulation, we consider a range of priors in the rare-event regime, i.e. $\pi_0^1 \in [0.05, 0.20]$. Finally, the miss and false-alarm costs are given by $C_M^1 = 2, C_A^1 = 1$ to emphasize that the miss risk is higher in this setting.

The dataset is input to each of the three analyses discussed above. The scalar output scores from each analysis are taken as its respective features, resulting in a total of three feature sets/groups/types. The discriminative power of each feature type, or equivalently the performance of an analysis, can be quantified using receiver operating characteristic (ROC) and precision-recall (PR) curves as shown in Fig. 6.4. From the figure, it is evident that the temporal-spectral feature is better than the spectral feature, which in turn is better than the generic energy feature, at detecting GCW calls.

The conditional probability mass functions (PMF), i.e. $\mathrm{p}_i(y_i|x)$, of features from each analysis can be estimated up to some quantization level, i.e. 100. Furthermore, as alluded to in Section 6.2.1, energy-based and spectral-based features, by construction, are inadequate to characterize GCW calls, and hence there are inherent uncertainties in these features for the detection of GCW calls. These uncertainties can be explicitly accounted for in the features' distributions using the uncertainty model discussed in Section 6.2.1,

[6] Profiled in MatLab for ML100G-10.

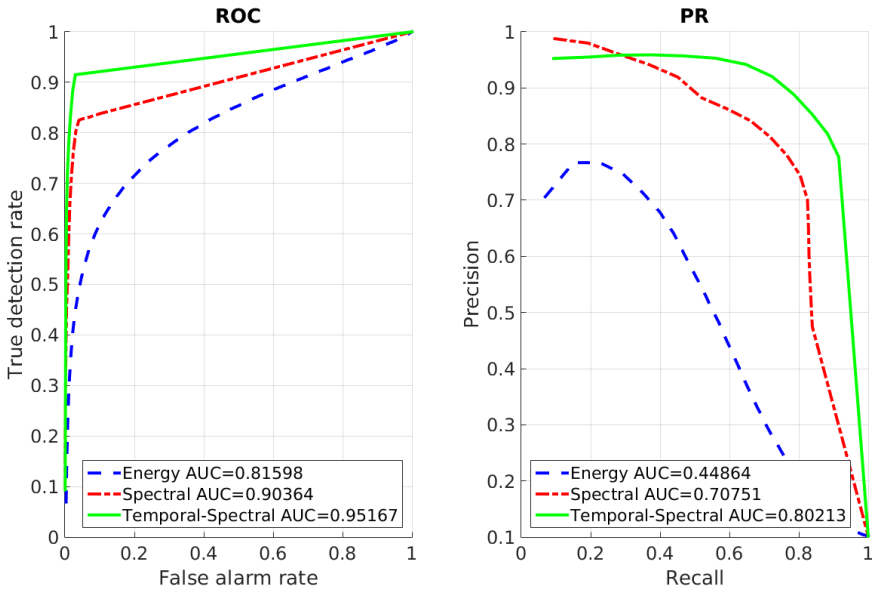


Figure 6.4: Receiver operating characteristic (ROC) curves and precision-recall (PR) curves of the features produced by the 3 analyses. Same as Fig. 4.6.

with the following parameters.

$$\begin{aligned} \epsilon_{01}^{1} &= \epsilon_{02}^{1} = 0.1 \\ \epsilon_{11}^{1} &= \epsilon_{12}^{1} = 0.1 \\ \nu_{01}^{1} &= \nu_{02}^{1} = 0.1 \\ \nu_{11}^{1} &= \nu_{12}^{1} = 0.1 \end{aligned} \tag{6.17}$$

Intuitively, the ϵ and the ν parameters indicate the level and the strength of a contamination on the nominal distribution, respectively. A formal method to set these parameters is left for future work. Finally, it is assumed that the temporal-spectral analysis (the last stage) is sufficient to characterize GCW calls and hence there is no uncertainty in this feature set.

6.3.2.2 Secondary application

To illustrate the benefit of feature sharing, we invoke the following twin-comparison argument. We consider a hypothetical secondary application that is, as far as the resource-performance trade-off is concerned, identical to the primary application. Namely, all parameters, such as the resource cost

and the feature models at each stage, of the secondary application are the same as those of the primary one. Due to the asymmetry in feature sharing between the primary and secondary applications, it is expected that there will be differences in the resulting resource consumption and detection performance of the two applications, and the merit of the proposed feature sharing approach can be evaluated by quantifying this difference. It is worth noting that this experiment represents the best-case scenario of feature-sharing, while the worst-case one is simply where the total resource consumption and detection error is the sum of both applications.

6.3.3 Results

The method developed in Section 6.2 can be used to optimize the acoustic system and the results are presented below.

Figure 6.5 breaks down the primary application's risk into the weighted resource consumption, and the miss and false-alarm rates to provide an intuitive understanding of the optimal policies. Furthermore, the average resource consumption of the primary application across all priors is 44.406 mJ (per audio frame). Without feature sharing, it is expected that the resource consumption of the secondary application would be the same as that of the primary one. However, the average resource consumption of the secondary application across the priors of interest can be found to be 4.877 mJ (based on the break-down of the secondary risk illustrated in Fig. 6.6), which is approximately a 9× reduction in resource consumption. This significant resource-saving is due to the fact that extracted features are shared and not recomputed among applications. In addition, the average detection risk (both the miss and false-alarm rate) of the secondary application is also reduced by 1.43× (from 4.75% to 3.31%). This reduction in risk illustrates that using shared features can be more beneficial than having no feature at all.

It is worth noting that the above applications' resource consumptions are the result of setting the regularization parameter λ to 0.0043, which is optimal for a resource/energy budget of 49.398 mJ (the sum of power consumptions by both applications and the 3.6×0.032 mJ base line from the microphone, the preamp circuit, and the ADC of the sensor over a frame). For a given

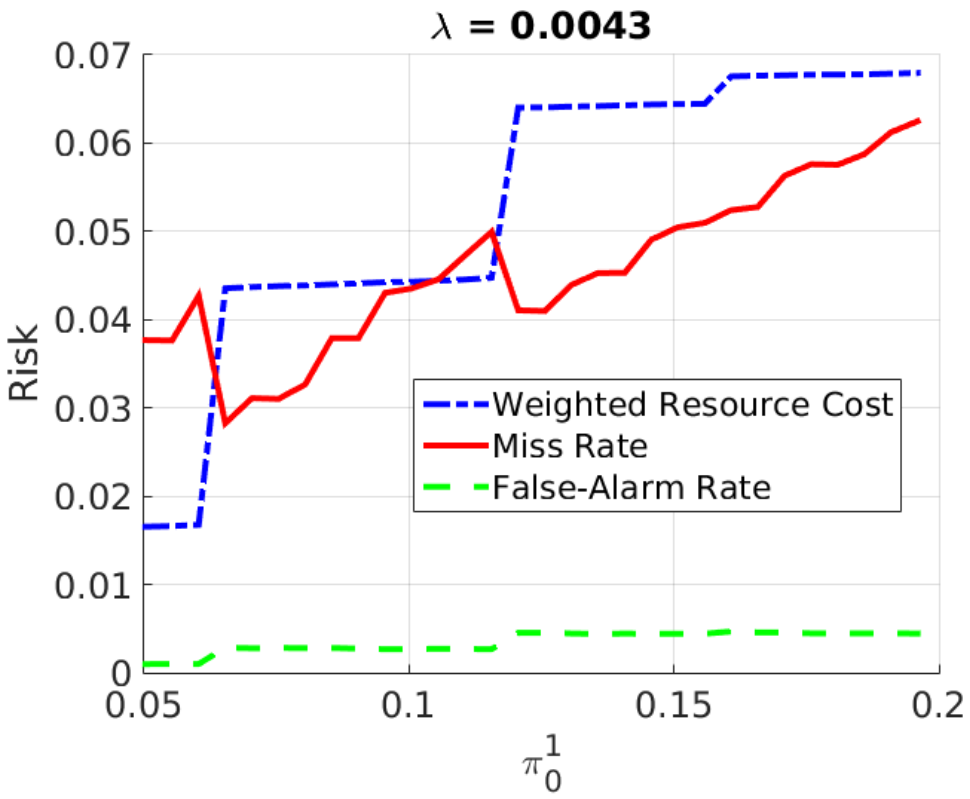


Figure 6.5: Breakdown of the optimal primary risk into components (see Eq. (6.4)): false negative (miss), false positive (false-alarm), and Lagrangian-weighted resource consumption. Low false-alarm rate is achieved across the priors of interest. The miss rate tends to increase with the prior. At a certain level, the primary application must ramp up its resource consumption or incur more false-alarms to reduce the miss rate.

resource budget, one needs to solve for λ. Numerical solution of the scalar variable λ is straightforward and hence shall be skipped here.

The primary and secondary decision rules are illustrated in Figs. 6.7 and 6.8, respectively. Note that the optimal policies of the secondary application take advantage of feature sharing whenever possible. Namely, $\delta_i^{2*} = F^1$ for all π_i^2 and $\pi_i^1 \geq \tau_i^{1*}$, $i = 0, 1, 2$. When the primary feature is not available, the secondary policies choose between extracting the secondary feature ($\delta_i^{2*} = F^2$) or making an early negative decision ($\delta_i^{2*} = 0$).

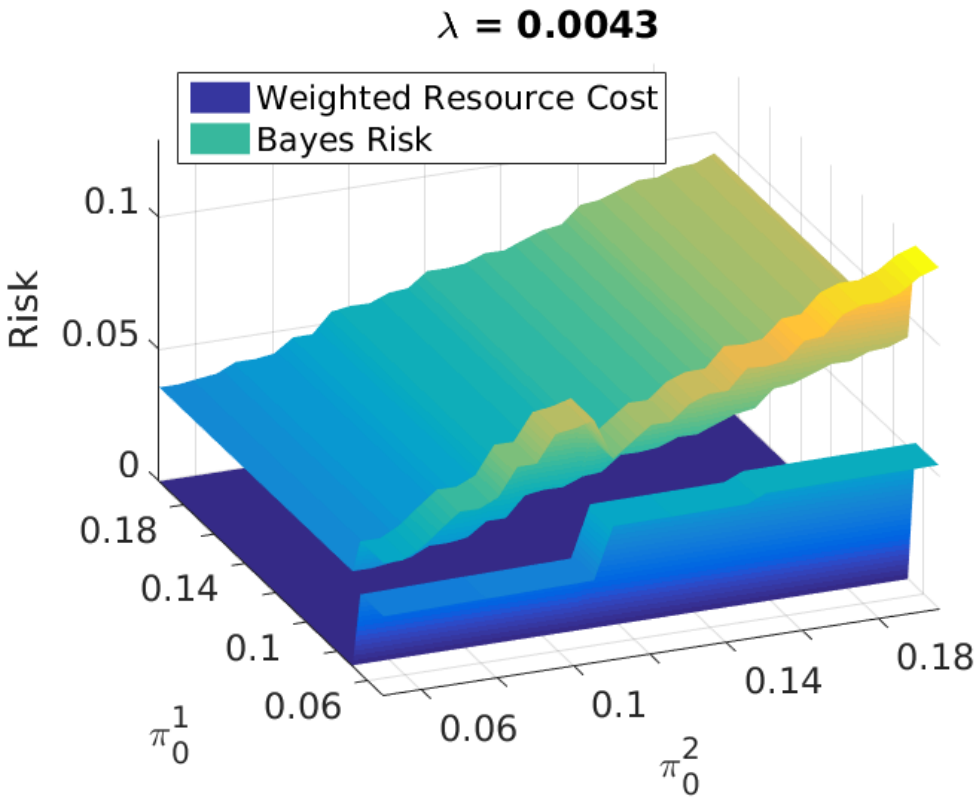


Figure 6.6: Breakdown of the optimal secondary risk into components (see Eq. (6.4)): Detection risk and Lagrangian-weighted resource consumption. The detection risk tends to increase with the secondary prior. At a certain level, the secondary application must ramp up its resource consumption to reduce the risk.

6.4 Summary and conclusions

This chapter investigates and shows the benefits of feature sharing in the optimization of resource-performance trade-off for detection systems with multiple applications. The proposed system model focuses on the vertical design, rather than the horizontal one commonly seen in the wireless sensor network literature. Namely, it is assumed that there is only one device at each layer/stage in the system stack, as opposed to having multiple devices at the same layer. Therefore, this work can complement prior works and vice versa. For instance, Chamberland et al. showed that it is asymptotically optimal for all sensors in a power-constrained sensor network to adopt the same (transmission) strategy, as the number of sensors in the network goes to infinity [39]. The result in [39] therefore allows ours to be applicable for

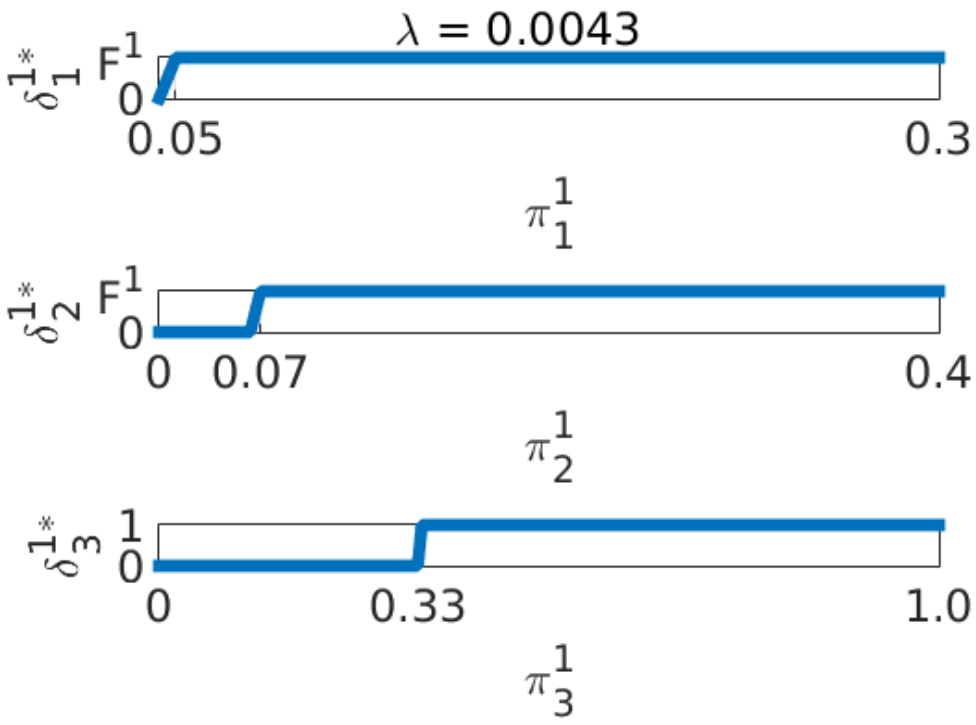


Figure 6.7: Optimal decision rules of the primary application $\delta_i^{1*}(\pi_i^1) \in \{F^1, 0, 1\}, i = 1, \ldots, 3$.

layers with more than one device. Finally, a proof-of-concept implementation of the acoustic system in Section 6.3 (with the sensor as an Android app) is available online for demonstration.[7]

[7] At `http://acoustic.ifp.illinois.edu`

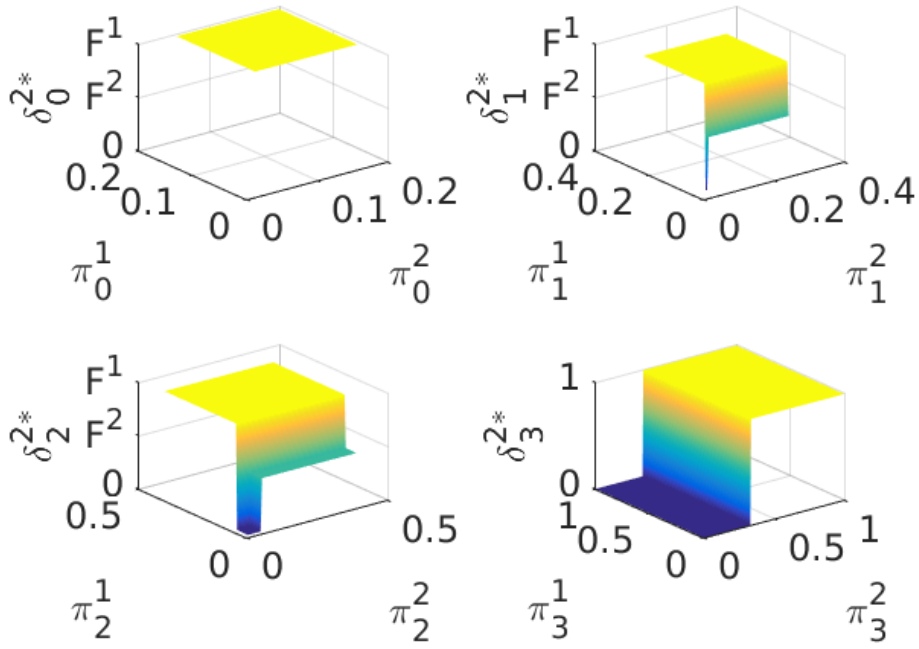


Figure 6.8: Optimal decision rules of the secondary application $\delta_i^{2*}(\pi_i^{1:2}) \in \{F^1, F^2, 0, 1\}, i = 0, \ldots, 3$.

CHAPTER 7

INTERFERENCE-ROBUST AUDIO CLASSIFICATION WITH LIMITED TRAINING DATA

7.1 Overview

Automatic audio classification is needed in many applications, especially in biological studies of wildlife. However, existing approaches (e.g. deep neural networks, hidden Markov models, etc.) fail to meet the expectation of practical applications due to the following reasons. First, to achieve good performance, these techniques often require a large amount of labeled data, which is generally unaffordable for most applications [99, 100] (with some exceptions such as English speech recognition [101]). In addition, while many of these techniques can handle noisy data [99], they do not account for *interference* (which usually presents in field recordings, e.g. sounds from a different species interfering with the target one). Hence, their performance is expected to degrade drastically under such conditions, as will be demonstrated in Section 7.4.

Motivated by the above observations, an improved audio classification technique is proposed to be interference-robust with limited (< 10) training data/templates.

- A sparse, AI-gram-based [102] time-frequency representation (TFR) is proposed to ensure noise-robustness. In addition, this universal procedure is lightweight, making it also suitable for on-sensor (even fixed-point) implementation.

- A multidimensional dynamic time warping (DTW) algorithm is proposed as a principled approach to handle limited training datasets. In addition, the algorithm utilizes a pseudo-cosine similarity, which naturally leads to the geometric-mean (GM) being used in template fusion for interference-robustness.

These ideas will be articulated further in Section 7.3.

The rest of this chapter is organized as follows. Section 7.2 surveys recent works on DTW-based algorithms in audio, while the proposed technique to improve upon existing literature is given in Section 7.3. Various experiments designed to test the robustness of the proposed approach are described in Section 7.4, followed by conclusions in Section 7.5.

7.2 Related works

The DTW algorithm can align two time series with different lengths and is therefore a core routine in solving both the template fusion and matching problems.

7.2.1 Template fusion

In field recordings, it is not uncommon that data are corrupted by noisy and/or interferences. Given a limited set of labeled data/templates, the *template fusion* problem seeks to combine/fuse them to form a clean one, which can then be used in subsequent *template matching*.

Kaewtip et al. approached this problem by proposing an ad-hoc spectrogram fusion algorithm [100]. Namely, given $M > 2$ templates, one can be chosen as a target whose length is desirable for the rest. The standard DTW algorithm is used to align each of the $M-1$ templates to the chosen one, i.e. $M-1$ pairwise alignments. A cleaner template can then be formed by taking the median of all aligned templates. The process might need to be repeated multiple (3 in [99]) time until an acceptable result is obtained. Note that there is no convergence guarantee.

7.2.2 Template matching

Template-matching is a well-known approach for audio classification. The method works by employing the DTW algorithm [103] to *align* a template and a test signal[1] so that their frame-by-frame similarity can be computed.

[1]Usually, a template is warped to align (have the same length) with a test data to streamline evaluation. However, this is the opposite of the convention used in [100].

The closeness of the two time series is then taken as the sum of per-frame similarities. Compared to alternative approaches such as hidden Markov models (HMM), DTW has better performance under the limited data regime [100]. Hence, it is chosen to be the basis for further improvement in this work.

DTW is merely a principled procedure to align two time series, and the actual performance of a DTW-based classifier depends heavily on the chosen signal representation. Early DTW-based classifiers use the spectrogram representation directly, which suffers from noisy data [99]. Kaewtip et al. addressed this problem by using both a time-frequency mask and appropriate weighting to emphasize *prominent (high-energy) regions* in the spectrogram [99]. It is empirically demonstrated that the prominent-region DTW outperforms the HMM approach, the standard (spectrogram only) DTW approach [100], and naive/generic machine-learning approaches. Namely, vectorized spectrographic images are used as feature vectors, with principal component analysis (PCA)-based dimensionality reduction, followed by either a support vector machine (SVM), a sparse-representation-based (SR-based), or a nearest-subspace-based (NS-based) classifier [104, 99]. There are also reports of prominent-region DTW classifier being used as a component in larger systems. For instance, a system of a DTW-based detector and an SVM classifier in tandem was proposed in [105] for joint segmentation and classification. Tan et al. proposed a system of one DTW-based and two SR-based classifiers to achieve synergy in performance [106].

An important insight from Kaewtip et al. [100] is that by focusing on prominent spectrographic regions, which remain invariant under noise, a DTW-based algorithm can achieve noise-robustness. However, the procedure to find the prominent region in [100] is suboptimal, i.e. a per-frame threshold whose value is 0.2 of the maximum amplitude (in that frame) is used to find prominent (frequency) bins. Namely, temporal structures in prominent regions, which could significantly improve the reliability of their identification/detection by considering multiple frames for decision-making, are not exploited. For instance, during silence intervals in between signaling periods, it is evident that any prominent region detected by the above greedy procedure is spurious (see Fig. 7.7).

There are well-known techniques to exploit temporal structure in signals. For instance, the celebrated McAulay-Quatieri (MQ) sinusoidal tracker [29, 30] is the first to track sinusoids over time using a greedy approach. Kwok

and Jones [31] introduce signal-optimal instantaneous frequency (IF) estimation and tracking using an adaptive short-time Fourier transform (STFT) and an HMM-based dynamic program. Dubois et al. used a particle-filter to track time-varying harmonics [32]. However, there are also drawbacks in these techniques. Namely, the tracker in [31] can only track the most significant frequency component; the particle-filter-based solution in [32] is computationally intensive; and the classical MQ tracker in [29, 30] uses a suboptimal, greedy approach.

7.3 An AI-gram-Based, Multi-Dimensional Dynamic Time Warping Algorithm

7.3.1 AI-gram-based time-frequency representation

A TFR that uses the AI-gram [102] to preserve signal invariance is proposed to replace and improve upon the prominent region concept. Unlike the spectrogram, the AI-gram directly represents the (log-) signal-to-noise ratio (SNR) and is well motivated by human audio perception [102, Fig. 2]. In addition, a recursive procedure is proposed to exploit temporal structures for reliable detection of prominent regions.

Let $S(f,t)$ be a spectrographic image of an audio signal with T_{blk} sample-per-frame and T_{inc} sample increment, where $0 \leq f < F$ and $0 \leq t < T$ are the frequency and time indices, respectively. Then, the noise floor $n(f,t)$ is multiplicatively tracked as follows:

$$n(f,t+1) = \begin{cases} n(\mathcal{F}(f),t) \times r_{\text{up}}, \text{ if } S(f,t) > n(f,t), I(f) < 0 \\ n(\mathcal{F}(f),t) \times r_{\text{upSlow}}, \text{ if } S(f,t) > n(f,t), I(f) \geq 0 \\ \max(n_\epsilon, n(\mathcal{F}(f),t) \times r_{\text{down}}), \text{ if } S(f,t) \leq n(f,t) \end{cases} \tag{7.1}$$

where $\mathcal{F}()$ is the (hash) mapping from the current to the previous frequency index to be used in the recursion, and shall be defined later. $r_{\text{up}}, r_{\text{upSlow}}, r_{\text{down}}$ are the multiplicative ratios to either ramp up or down the noise floor, and are set at $1.01, 1.005, 0.99$ in the current implementation, respectively. n_ϵ is the assumed minimal possible noise floor. Notice that the noise floor is increased

more slowly in the beginning, since it is more likely that the amplitude rise is due to signal being present, and returns to the normal incremental rate afterward. This slow-start period is defined by the following indicator:

$$I(f) = \begin{cases} I(f) - 1, S(f,t) > n(f,t) \\ T_\tau, \text{ else} \end{cases} \tag{7.2}$$

where T_τ is a time constant that is set proportional to the duration of the longest (uninterrupted) signal component in the target.

With the noise floor, the spectrogram can be converted into an AI-gram-based representation as follows:

$$\text{snr}(f) = \max(0, \log(\frac{S(f,t)^2}{n(f,t)^2})) \tag{7.3}$$

Note that there is a minor modification from the original AI-gram of [102], wherein there is no upper cap of 1.

Next, the following recursive update is used to ensure temporal structures are accounted for. Let

$$\begin{aligned} f_{\text{low}}(f) &= \max(0, f - f_{\text{off}}) \\ f_{\text{high}}(f) &= \min(F - 1, f + f_{\text{off}}) \end{aligned} \tag{7.4}$$

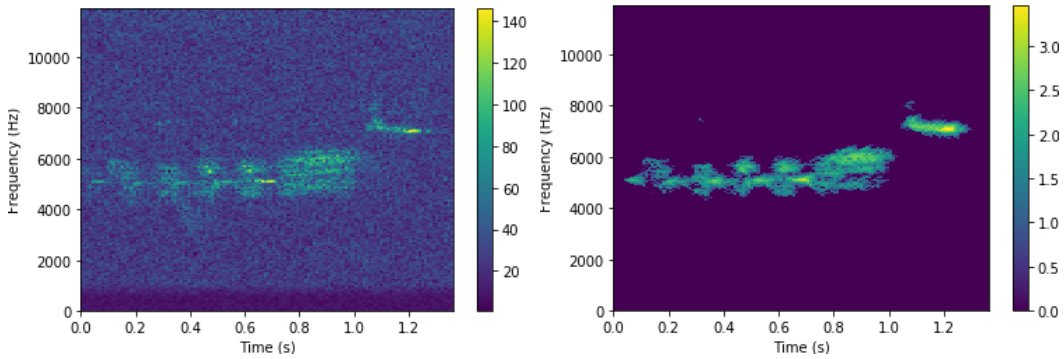
be the spectral boundaries within which temporal structures are to be exploited. f_{off} is the frequency offset, and all indices are assumed to start at 0. Then

$$\text{winFun}(f'; f) = 1 - 0.05\frac{|f - f'|}{f_{\text{off}}}, f' \in [f_{\text{low}}, f_{\text{high}}] \tag{7.5}$$

is a bell-shaped window function over the interval $[f_{\text{low}}, f_{\text{high}}]$ and

$$\begin{aligned} \text{objFun}(f'; f) &= \alpha \cdot \text{snrAcc}(f') + \text{winFun}(f'; f) \cdot \text{snr}(f), f' \in [f_{\text{low}}, f_{\text{high}}] \\ \text{snrAcc}(f) &= \max_{f'}(\text{objFun}(f'; f)) \\ \mathcal{F}(f) &= f_{\text{low}} + \arg\max_{f'}(\text{objFun}(f'; f)) \end{aligned} \tag{7.6}$$

where $\text{objFun}(f'; f)$ is the recursion's objective function. α is the exponential forgetting factor that is related to the time constant T_τ, i.e. $\alpha = \exp(-\frac{1}{T_\tau})$. $\text{snrAcc}(f)$ is the accumulated $\text{snr}(f)$ function up to the current time, and is



(a) Spectrogram. (b) AI-gram-based TFR.

Figure 7.1: A sample transformation from spectrogram to AI-gram-based TFR.

updated by maximizing the objective function. Its maximizer, offset by f_{low}, defines the mapping $\mathcal{F}(f)$. Namely, the recursion in (7.6) ensures that the (log-)SNR is maximized along the entire time-constant T_τ, and not just at each frame.

Depending on whether the target signal belongs to the animal vocalization class (which is characterized as narrow-band and long duration) or non-biological environmental sound class (which has broadband and short duration) [107], an optional max-pooling operation on a spectral frame can be executed to optimize the representation for the former.

Lastly, the AI-gram TFR, denoted by X, is obtained by suppressing (with subtraction) $\text{snrAcc}(f)$ for some threshold γ, to induce sparsity.

$$\text{X}(f,t) = \begin{cases} \text{snrAcc}(f) - \gamma, & \text{if } \text{snrAcc}(f) > \gamma \\ 0, & \text{else} \end{cases} \tag{7.7}$$

This concludes the conversion from the spectrogram $S(f,t)$ to the AI-gram-based TFR $X(f,t)$. The complete algorithm is given in Appendix C.1 and a sample transformation is shown in Fig. 7.1. The remainder of this chapter assumes all data use the representation presented in this section.

Finally, it is worth noting that this universal procedure is lightweight, and thus suitable for implementation at the edge of the cloud, i.e. on battery-powered sensors, for acoustic event detection. Specifically, the Android-based sensors of the proposed sensing service (see Section 2.2.2) implement a slightly modified version of this algorithm, in which a spectral-based max-

pooling operator is executed just before the final suppression (in Eq. (7.7)) to produce prominent TF ridges, which can be transmitted with a reduced bandwidth rate compared to TF regions.

7.3.2 Multidimensional dynamic time warping

Compared to [100], a more principled approach for template fusion is given in this section. From a set of M templates, a template with desirable length is chosen. Then all $M-1$ templates are aligned to the chosen one in a single pass, using the proposed *multidimensional DTW* algorithm. Once aligned, a natural choice for the fusion function is the geometric mean. Details are given as follows.

Cost tensor: Recall that the standard DTW aligns two time series by finding the shortest path through their cost matrix, i.e. its elements (or equivalently nodes) are frame-by-frame cost, defined as one minus the cosine similarity [103]. The multidimensional DTW generalizes the alignment process for M time series by finding the shortest path through their *cost tensor*, i.e. a multidimensional array, whose elements are also frame-by-frame cost. A major challenge with the multidimensional DTW procedure is that it is infeasible to find the shortest-path by inspecting the entire cost tensor due to its size. For instance, with only 10 templates, and each with an average length of 100, the number of nodes in the tensor is already 100^{10}!

To find a good suboptimal path with modest exploration of the cost tensor, the A* algorithm [108] can be employed. In particular, the static ϵ-weighted A* algorithm [109] is employed to ensure that the suboptimal path found by A* algorithm is within $\epsilon\times$ of the optimal one [110] (see Listing 7.1). The algorithm is an extension of the celebrated Dijkstra's algorithm that introduces a surrogate objective function, which is the sum of to the actual cost accumulated from the start and a heuristic function. The heuristic function estimates the remaining cost-to-go to reach the goal, which helps sidestep the need to visit (and hence evaluate) all elements in the cost tensor. In summary, let $\{t_i : 0 \leq t_i < T_i, i = 0, \ldots, M-1\}$ (or simply $\{t_i\}$), be a list of the templates' (time) frame indices, which also specifies a node in the cost

tensor. T_i denotes the length of template i in a set of M templates. Then

$$\underbrace{f(\{t_i\})}_{\substack{\text{surrogate cost from} \\ \text{the start } \{0\} \text{ to node } \{t_i\}}} = \underbrace{g(\{t_i\})}_{\substack{\text{actual cost from} \\ \text{the start } \{0\} \text{ to node } \{t_i\}}} + \underbrace{h(\{t_i\})}_{\substack{\text{estimate cost from} \\ \text{node } \{t_i\} \text{ to the goal } \{T_i - 1\}}} \tag{7.8}$$

Listing 7.1: A* with heuristic weighted by $\epsilon \geq 1$ in Python

```
def Astar(Xs, start, goal, eps):
  dist = {}
  prev = {}
  frontier = minpq()
  explored = set()

  frontier[start] = 0+eps*getHeuristic(start, goal, Xs)
  dist[start] = 0
  while len(frontier) > 0:
    node = frontier.pop()

    if node == goal:
      return getPath(node, prev), dist[node]

    explored.add(node)

    for ngb in getNeighbor(node, nTs):
      if ngb not in explored:
        if ngb not in frontier:
          frontier[ngb] = np.infty
          dist[ngb] = np.infty

        alt = dist[node] + getDist(node, ngb, Xs)
        if alt < dist[ngb]:
          frontier[ngb] = alt+eps*getHeuristic(ngb, goal, Xs)
          dist[ngb] = alt
          prev[ngb] = node

  return None, None
```

Heuristic function: The heuristic function $h(\{t_i\})$ is an estimate of the cost-to-go from node $\{t_i\}$ to the target node $\{T_i - 1\}$. Euclidean distance between nodes is chosen to ensure that $h(\cdot)$ is both *admissible* and *monotonic/consistent* in the A* sense [111].

Global constraint: There is no global constraint. Namely, unlike the standard DTW algorithm in which a global constraint is needed to limit the search space [103], it is herein naturally limited by the guidance of the heuristic function (see Fig. 7.8c).

Local constraint: The local constraint is a generalization of the Type I local constraint in [103], where each node in the cost tensor C has neighbors within 0, 1, or 2 steps away in each dimension, with their step difference limited to at most 1, and excluding the node that is 2 steps away in all dimensions. Namely, let $\mathcal{N}$ denote the neighbor set of node $\{t_i\}$, then

$$\mathcal{N}(\{t_i\}) = \Big\{\{t_i+\delta_i\} : \delta_i \in \{0,1,2\}, \max_{k,l} |\delta_k - \delta_l| \leq 1\Big\} \backslash \Big\{\{t_i\} \cup \{t_i+2\}\Big\} \quad (7.9)$$

Figure 7.2 illustrates the local constraint for $M = 2$.

Similarity function: The similarity metric is the pseudo cosine similarity (correlation coefficient), which is a natural generalization of the standard one between spectral vectors $X_i(\cdot, t_i)$ of each template indexed by i, i.e.

$$s(\{t_i\}) = \frac{\sum_{f=0}^{F-1} \Pi_{i=0}^{M-1} X_i(f, t_i)}{\Pi_{i=0}^{M-1} \|X_i(\cdot, t_i)\|} \quad (7.10)$$

where the $(\cdot, t_i)$ denotes all (frequency) components, i.e. a spectral vector, at time t_i. Note that this metric is scale-invariant, i.e. amplifying X_i does not change the similarity measure.

Let $g(\{t_i\})$ denote the distance from the start node, i.e. $\{0\}$, to node $\{t_i\}$. Then the distance to a neighbor of $\{t_i\}$, denoted by $\{t_i'\}$, is given by

$$g(\{t_i'\}) = \begin{cases} g(\{t_i\}) + 1 - s(\{t_i'\}), \text{ if } \{t_i'\} \in \mathcal{B}(\{t_i\}) \\ g(\{t_i\}) + 1 - \min(1, s(\{t_i'\}) + s(\{t_i' - 1\})), \text{ otherwise} \end{cases} \quad (7.11)$$

where $\mathcal{B}$ denotes the unit box (i.e. radius 1 in all dimensions).

Once all (corrupted) templates are aligned, they can be combined/fused to remove interferences and form a cleaner one. This is because the optimal warp path output by the DTW procedure maximizes the alignment of com-

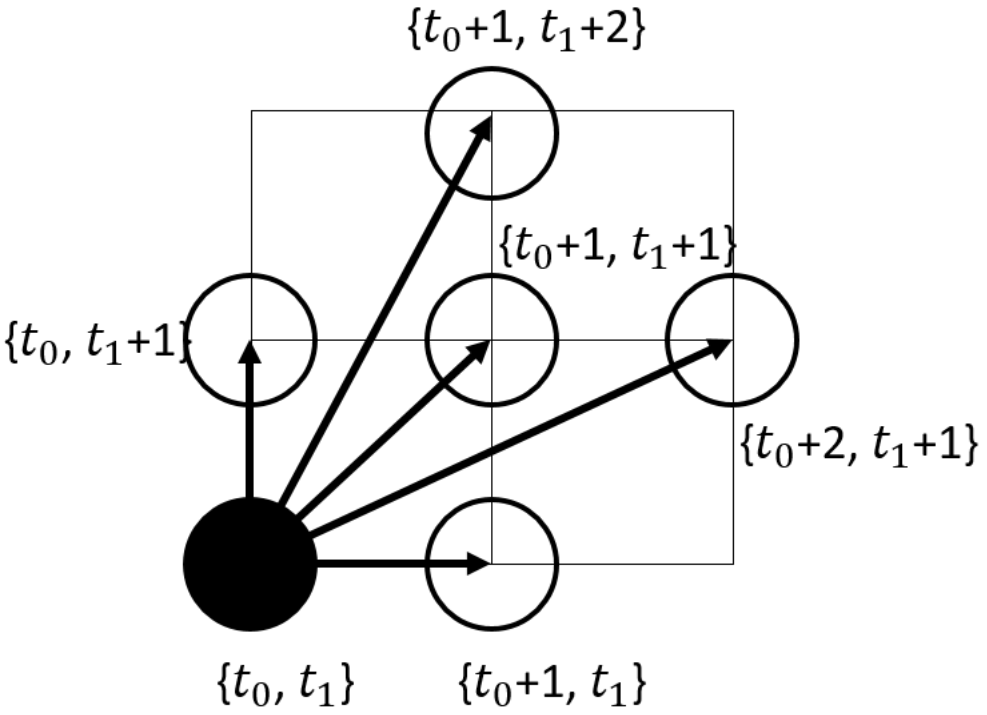


Figure 7.2: Illustration of the local constraint at node $\{t_0, t_1\}$ for $M = 2$. The circles are valid neighbors of the black node.

mon, most likely signal, components in all templates. Namely, let i_0 be a target template index whose length is desirable for the fused template X^*. Then X^* can be constructed as follows.

$$\begin{aligned} X^*(f,t_{i_0}) &= \sqrt[M-1]{\Pi_{i\neq i_0} X_i(f,t_i)},\ \text{if } t_{i_0} - t'_{i_0} = 1 \\ X^*(f,t_{i_0}-1) = X^*(f,t_{i_0}) &= \sqrt[M-1]{\Pi_{i\neq i_0} X_i(f,t_i)},\ \text{if } t_{i_0} - t'_{i_0} = 2 \\ X^*(f,t_{i_0}) &= \left[X^*(f,t_{i_0}) + \sqrt[M-1]{\Pi_{i\neq i_0} X_i(f,t_i)}\right]/2,\ \text{if } t_{i_0} - t'_{i_0} = 0 \end{aligned} \tag{7.12}$$

where $\{t_i\}$ and $\{t'_i\}$ denote an arbitrary node and its preceding neighbor on the optimal warp path. From the similarity metric in Eq. (7.10), the geometric mean is a natural choice for the fusion function, because X^* will yield the same similarity measure (up to a scaling factor) as those it replaces, i.e.

$$\underbrace{\sum_{f=0}^{F-1} \Pi_{i=0}^{M-1} X_i(f,t_i)}_{\substack{\text{Numerator in the} \\ \text{pseudo-cosine similarity}}} = \sum_{f=0}^{F-1} X_{i_0}(f,t_{i_0}) X^*(f,t_{i_0})^{M-1} \tag{7.13}$$

Biological motivations for this multiplication-based processing can also be found in established neuron studies [112, 113].

7.4 Experimental results

The performance of the proposed algorithm in Section 7.3 is evaluated here on both a controlled dataset with various (stationary and transient) interference across a wide range of signal-to-interference ratio (SIR), and an actual field recording for the detection of Golden-cheeked Warbler (GCW)'s Type-A calls [95]. Since the GCW is an endangered bird species, this application has important implications for its conservation. The baseline for comparison is the current state-of-the-art method by Kaewtip et al. [99, 100], which itself outperforms others including HMM, standard DTW, SVM, SR, and NS.

Throughout these experiments, all spectrographic analysis is carried out with a 21 ms window with 11 ms increment, which are power-of-two approximations (for $T_{\text{blk}}, T_{\text{inc}}$, respectively) to the desired 32 ms window with 16 ms increment. The Hann window, which has strong and smooth sidelobe suppression, is chosen (instead of the standard Gaussian windows for the

minimal time-frequency uncertainty [114]) to smooth the tracked noise floor. The sampling frequency $f_s = 24$ kHz and 512-length FFT are used. The time constant is set to be 32 ms, i.e. $T_\tau = \frac{32e-3}{T_{\text{inc}}}$ rounded to the nearest integer. Lastly, the frequency offset f_{off} is set to 2 so that only chirps with an admissible rate of $2\frac{f_s/512}{T_{\text{inc}}} = 8.5$ Hz/ms or less are tracked.

7.4.1 Synthetic interferences

There are two types of synthetic interference considered: stationary and transient. Stationary interference is made up of sinusoids generated with either constant or time-varying frequency (i.e. logarithmic-chirp). In addition, these can either be present for all or half of the signal duration and might or might not overlap in time-frequency with the target signal. Transient interference is made up of two song phrases from another bird species, i.e. Cassin's vireos (Vireo cassinii; CAVI) [115]. Table 7.1 summaries all different types (10 in total) of interference. Figure 7.3 shows the spectrogram of interfered templates at 0 dB SIR, which is defined as the ratio (in dB) between the maximum power of the signal and interference's TFR.

Table 7.1: Experimental interference types.

Interference types	Description
1	stationary, constant frequency, full-duration, non-overlapping
2	stationary, constant frequency, half-duration, non-overlapping
3	stationary, constant frequency, full-duration, overlapping
4	stationary, constant frequency, half-duration, overlapping
5	stationary, logarithmic-chirp, full-duration, non-overlapping
6	stationary, logarithmic-chirp, half-duration, non-overlapping
7	stationary, logarithmic-chirp, full-duration, overlapping
8	stationary, logarithmic-chirp, half-duration, overlapping
9	transient, CAVI's song phrase 1
10	transient, CAVI's song phrase 2

For each interference type in Table 7.1, the SIR is varied from -18 to 3 dB with a 3 dB increment. And for each SIR level, two cases are considered. Either the template is corrupted by interference, or the test data is (under both hypotheses), since it is very unlikely that the exact same interference occurs in both. The outcomes from both cases are combined to obtain the area under the receiver operating characteristic (ROC), which is herein used as the metric to compare methods.

Experimental results are given in Fig. 7.4, where it is evident that the proposed AI-gram-based approach outperforms the prominent-region one across all interference types and SIRs. It is worth noting the peculiar performance of the latter where better results are achieved *below* the 0 dB SIR point (better performance at higher SIR is trivial). A closer look at the prominent-region's test statistics in Fig. 7.5a reveals the reason for this. Namely, under the corrupted template setting at 0 dB SIR, the performance degrades drastically.

(When the test data is corrupted, the performance degrades gracefully with the decrease in SIR.) The problem is due to the greedy approach in [100], which *disfigures* the resulting signal mask as shown in Fig. 7.5d, which causes more harm than completely erasing (and hence ignoring) it as in Fig. 7.5b. Hence the prominent approach does not perform well if a template is corrupted. The proposed AI-gram-based approach does not have this problem.

7.4.2 Field recordings

The dataset is a 46-minute, 24 kHz audio recording taken in the field in Rancho Diana, San Antonio's city park. The dataset contains 206 GCW calls (manually identified and labeled), each of whose duration is approximately one second. In addition to GCW calls, the dataset also contains various interferences from other animals' vocalizations, time-varying wind noise, etc., since it is taken directly from a field recording. Precisely, the fraction of GCW calls in the entire dataset is 10.19%.

Six heavily-interfered templates are hand-picked to represent the worst-case scenario and to stress-test both methods.[2] These are shown in Fig. 7.6.

With the spectrogram fusion algorithm in [99], the fused template (which includes the median spectrogram, the binary mask, and the weight vector) is shown in Fig. 7.7. It is evident that some interference gets into the resulting template. Furthermore, since the algorithm in [99] was designed for phrases, i.e. with the assumption that there are no silence gaps within a template, the binary mask is very noisy during silence intervals.

With the proposed template fusion algorithm, the AI-gram TFR of the fused template is given in Fig. 7.8a. In the current implementation, it is experimentally found that $\epsilon = 2.3$ is needed to ensure fast convergence time (approximately a minute). The multidimensional DTW finds the smallest-cost hyper-path through a cost tensor from the start to the end nodes (see Fig. 7.8c). With GM fusion, it can be observed that all interference is removed, and while some components of the signal are also destroyed, it does not significantly degrade the performance of subsequent template-matching, as prominent TF regions of the target signal can still be visually recognized in

[2]Regardless, these are not uncommon in practice.

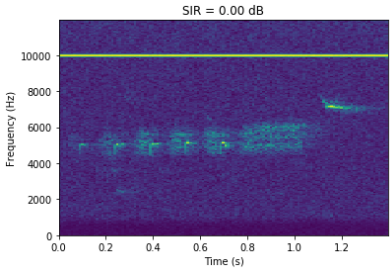
the warped template in Fig. 7.8d, and later proven with empirical evidence in Fig. 7.9.

The fused templates are window-slided along the field recording with 50% increment. Among overlapping windows, only the one with the maximal total score is retained. Finally, the per-frame template-matching score within each retained window is smoothed. The matching scores on the entire dataset are used as statistics to evaluate the ROC and precision-recall (PR) curves of competing methods. These are shown in Fig. 7.9. In either case, the proposed AI-gram-based method uniformly outperforms the other.

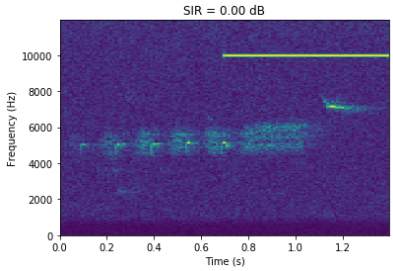
7.5 Summary and conclusion

This chapter shows that the combination of the AI-gram TFR of audio data and the multi-dimensional DTW procedure results in an audio classification technique that is interference-robust even with limited training data. For future work, the multi-dimensional DTW procedure could be improved to tighten the sub-optimality bound. For instance, anytime A* [116], which starts from a quick suboptimal solution but with loose bound, can progressively tighten the bound (while reusing previous search efforts) until terminated, then we have a promising candidate to achieve the task.

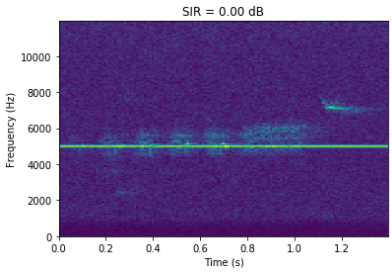
Moving further forward, the AI-gram TFR also provides a good starting point to develop concept-based learning algorithms similar to that found in [117]. Namely, Lake et al. proposed in [117] a more human-like approach for machines to learn, in which Bayesian thinking permeates, i.e. from the generation of (new) concepts from primitives, to the sampling of exemplars (and subsequently raw data) from generated concepts. The method in [117] outperforms the cumbersome deep-learning approach in many tasks, recognition among them, on handwritten characters, with only one training example. It is envisioned that the concept-learning approach, if adapted for the audio domain, could prove invaluable given the limited-training-data constraint.



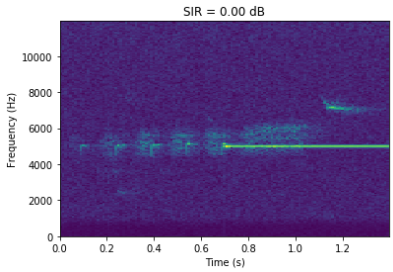
(a) Interference type 1.



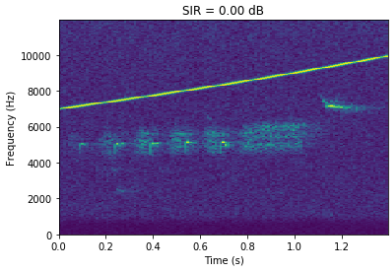
(b) Interference type 2.



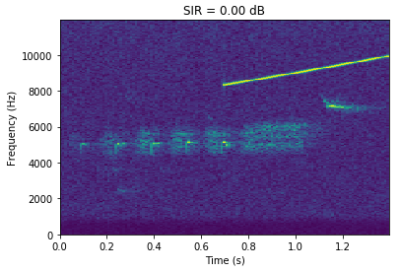
(c) Interference type 3.



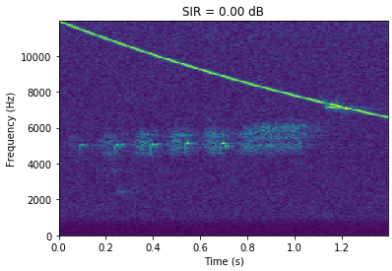
(d) Interference type 4.



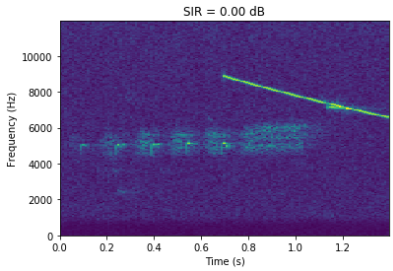
(e) Interference type 5.



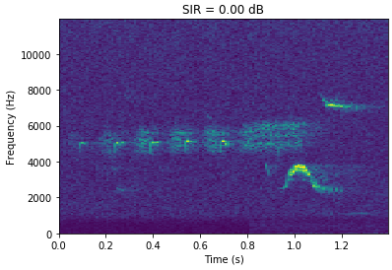
(f) Interference type 6.



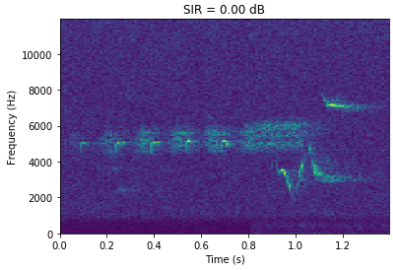
(g) Interference type 7.



(h) Interference type 8.

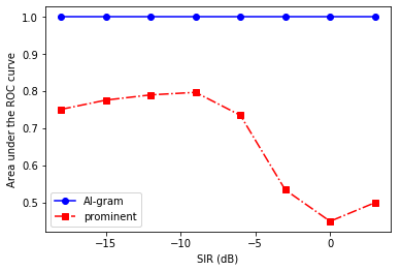


(i) Interference type 9.

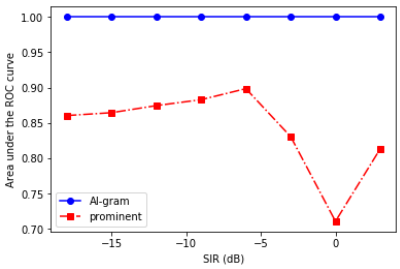


(j) Interference type 10.

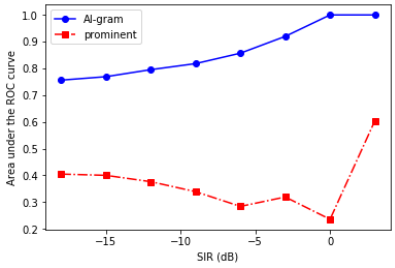
Figure 7.3: Illustrations of different interference types.



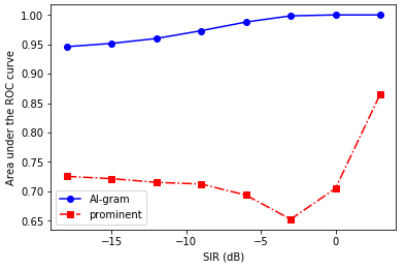
(a) Interference type 1.



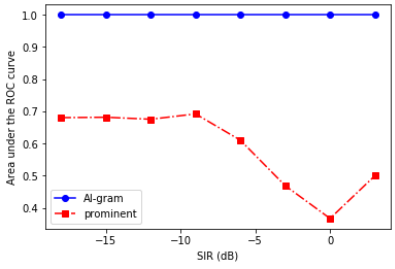
(b) Interference type 2.



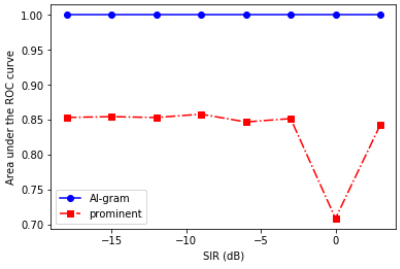
(c) Interference type 3.



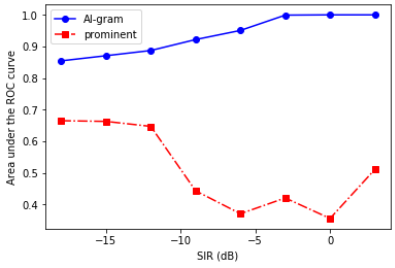
(d) Interference type 4.



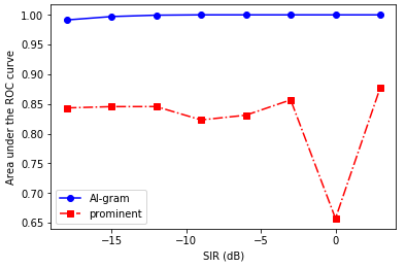
(e) Interference type 5.



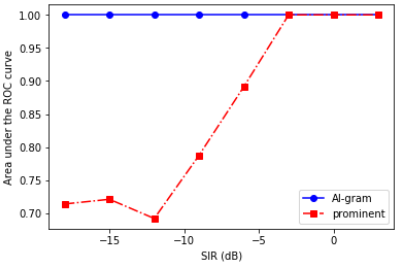
(f) Interference type 6.



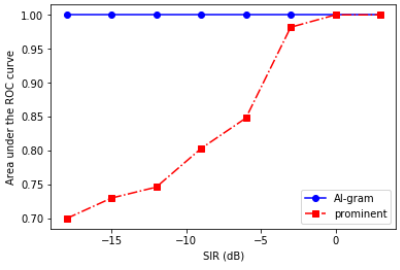
(g) Interference type 7.



(h) Interference type 8.

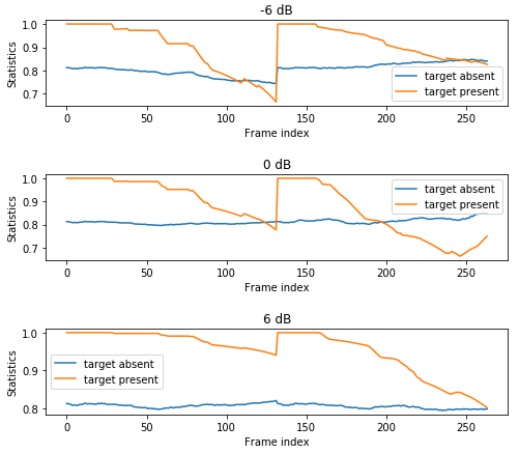


(i) Interference type 9.

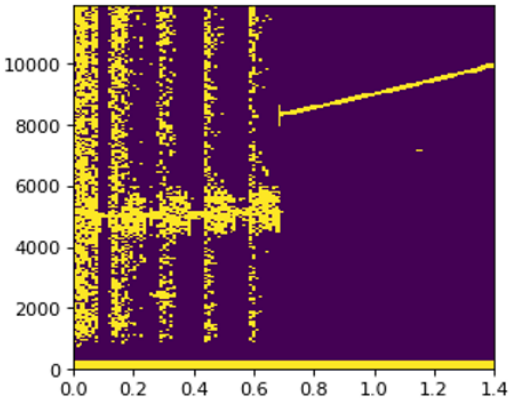


(j) Interference type 10.

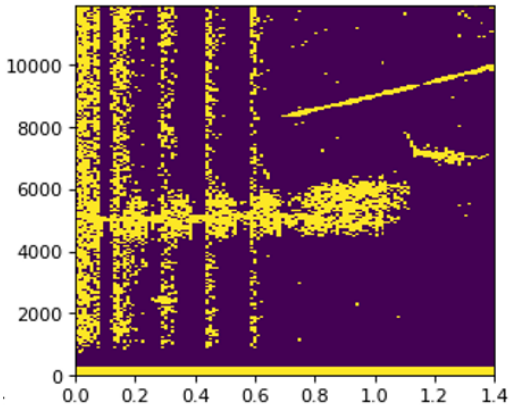
Figure 7.4: Area under ROC curves of all interference types.



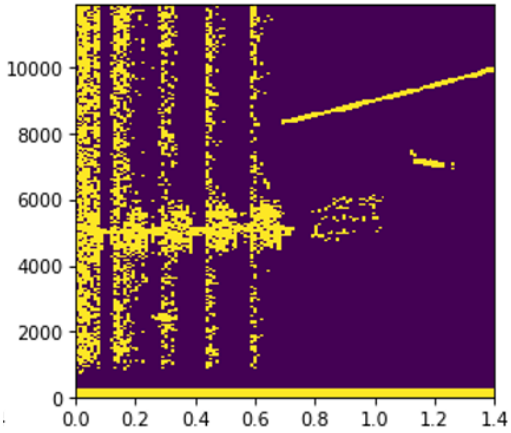
(a) The prominent region's test statistics with interference type 6 and SIR at -6, 0, 6 dB. In each subplot, the first half of the frames are under the corrupted test data setting, while the second half of the frames are under the corrupted template setting.



(b) The mask at -6 dB SIR.



(c) The mask at 6 dB SIR.



(d) The mask at 0 dB SIR.

Figure 7.5: Evidence explaining the poor performance at 0 dB SIR of the prominent-region approach. For a detailed explanation, the reader is referred to the last paragraph of Section 7.4.1.

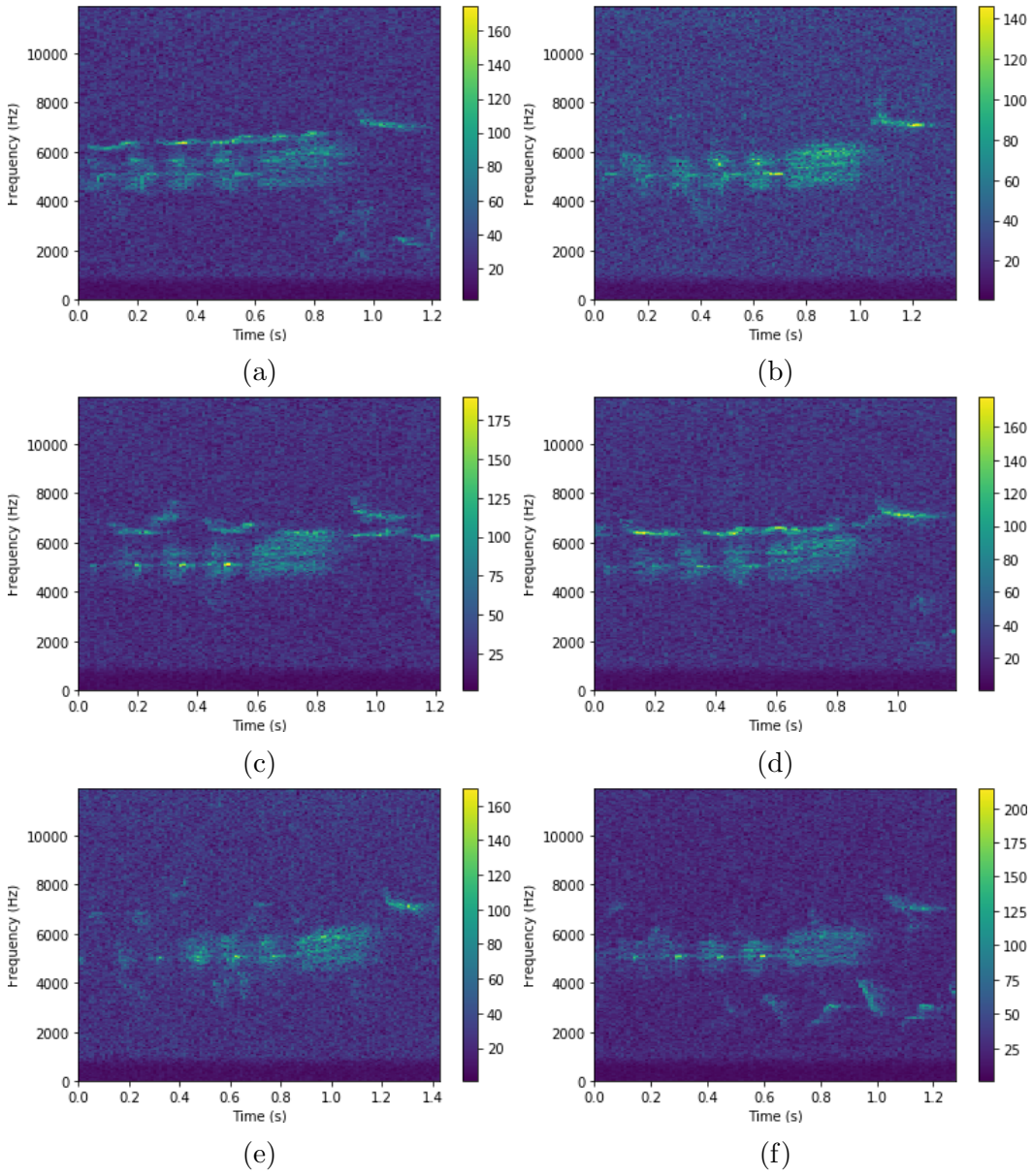
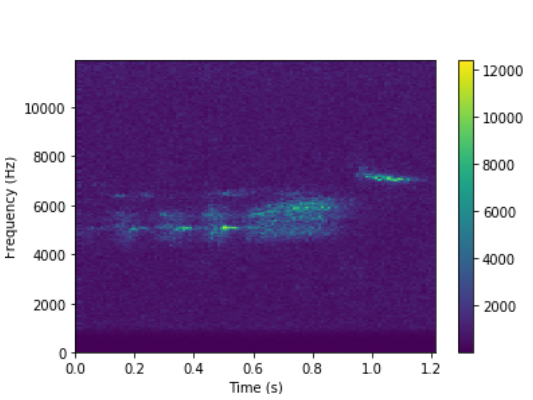
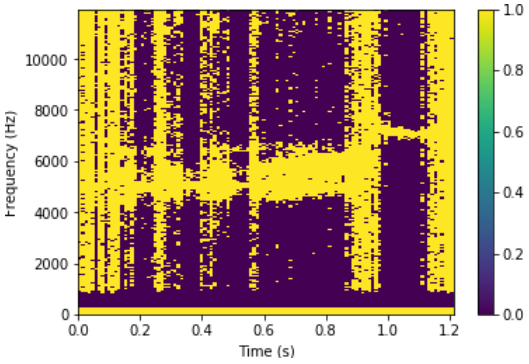


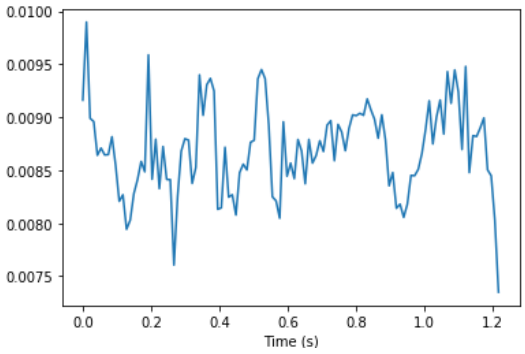
Figure 7.6: Strongly-corrupted templates chosen to represent the worst-case scenario. The duration of the templates are 116, 129, 115, 113, 135, 121 spectral (256 frequency points) frames.



(a) The median spectrogram.

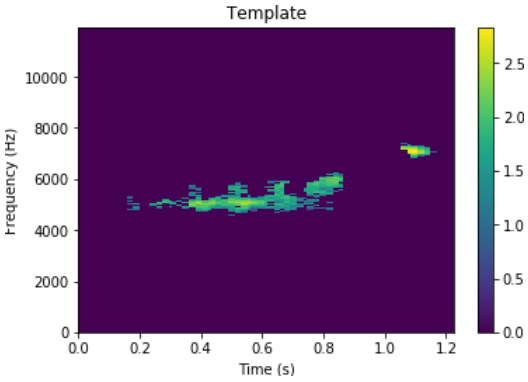


(b) The binary mask indicating prominent (high-energy) regions.

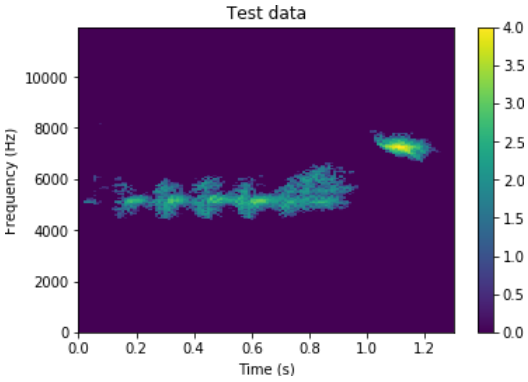


(c) The weight vector indicating prominent (high-energy) frames.

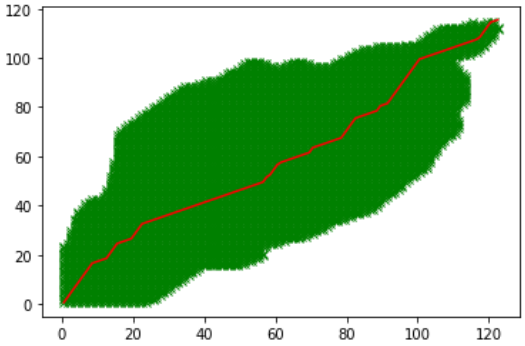
Figure 7.7: The fused template by the prominent-region DTW algorithm.



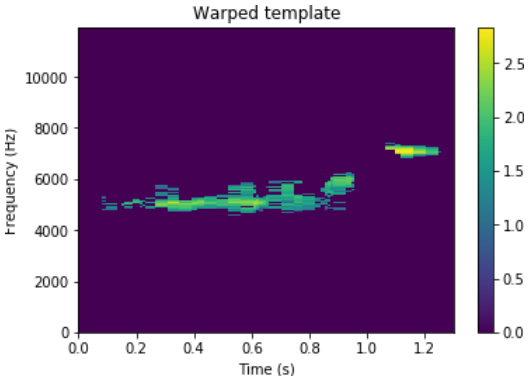
(a) The fused template.



(b) A test data.



(c) The cost matrix of warping the template to a test data. The optimal warp path is the red line within the explored (green) space.



(d) The warped template which aligns with the test data.

Figure 7.8: An example of template fusion and matching by the multi-dimensional DTW algorithm.

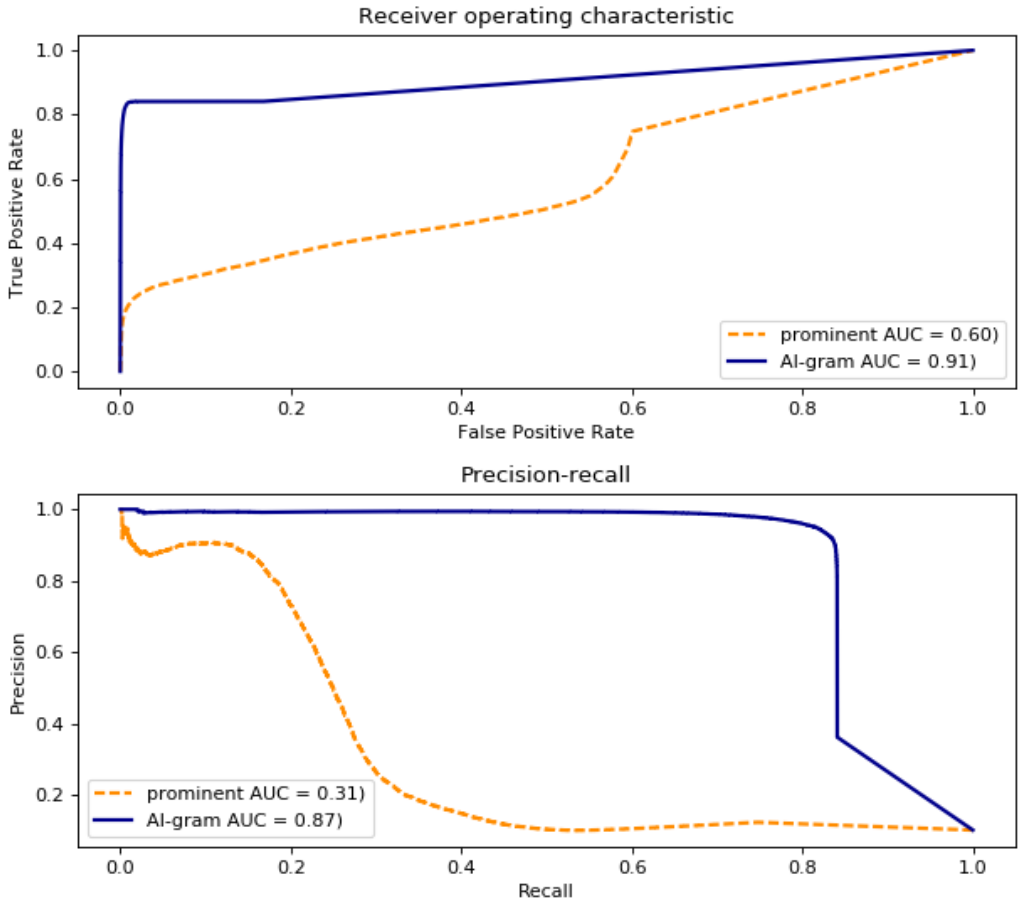


Figure 7.9: Comparison between the proposed (AI-gram) algorithm and the prominent-region-based method on field recordings. Note that the dip in precision of the prominent approach (near the low recall part of the curve) is evident that its test statistics can be misleading, i.e. not proportional to the true (but unknown) likelihood ratio, since the precision curve is not monotonically increasing with a threshold.

CHAPTER 8

CONCLUSIONS AND FUTURE WORKS

This work recognizes the gap between the resource-centric sensors and the inference(analytic)-centric applications of the IoT. While there exist other middle-wares that were proposed to bridge the gap, the sensing service introduced herein not only provides the necessary abstraction, but also has the advantage of being highly resource-efficient while embracing the general-purpose design philosophy needed to support various IoT applications. Resource-efficiency is achieved through the synergetic collection of novel contributions:

- A modular, data-centric architecture is proposed with a globally accessible database server, which serves both as a storage and communication medium for sensor and client modules.
- Resource management in sensing services is achieved using the guided-processing principle, which proposes that the execution of modules in a system depends on the inference results of prior modules. The proposed technique is provably more energy-efficient than the best possible system using the duty-cycling approach. Namely, on a practical detection application, the proposed approach significantly improves the detection performance (up to 1.7× and 4× reduction in false-alarm and miss rates, respectively) for the same energy consumption, when compared to the duty-cycling approach.
- Guided-processing is also generalized to graph-based systems.
- Sensing services need to support multiple applications, and feature-sharing is proposed as a way to achieve resource-efficiency in this setting. With a twin-comparison argument, it is shown that a system can achieve 9× resource saving and 1.43× improvement in detection performance using the optimal feature-sharing strategy.

- The proposed principles are used to build an audio sensing service prototype.

- For audio applications, it is recognized that an interference-robust audio classification technique with limited training data is needed. To this end, a novel algorithm is proposed, which combines AI-gram-based TFR and multidimensional DTW. The proposed technique outperforms the state-of-the-art on both field recordings (with areas under the receiver operating characteristic and precision-recall curves being 91% and 87%, respectively) and a controlled dataset with various interference across a wide range of SIR.

The above contributions prove that audio sensing in the IoT is feasible, given that resource-awareness is taken seriously. Furthermore, a prototype of an audio sensing service is built. The audio sensor module is implemented as an Android app, available for download at `https://play.google.com/store/apps/details?id=com.longle1.spectrogram`. The database server and various demonstrations of the audio sensing service are publicly available at `http://acoustic.ifp.illinois.edu`. Source code of the entire software package is also available under MIT license at the following repositories.

- The sensor module: `https://bitbucket.org/longle1/sas-sensor`.

- The database and web servers: `https://bitbucket.org/longle1/sas-servers`.

- The client module: `https://github.com/longle2718/sas-client`.

An interesting extension of this work is its application to video sensing in the IoT, which is evidently even more resource-constrained than audio. It is envisioned that due to the complexity involved with video processing, a practical graph-based sensing system will be needed, and thus will greatly benefit from the general guided-processing principle. Furthermore, it is noted that resource management in a video sensing service could take advantage of popular image quality assessment algorithms [118, 119], which offer an automatic and accurate mechanism to gauge the utility of an image frame. Such capability is evidently handy for evaluating the cost-performance tradeoff in a video sensing service, and its eventual optimization.

APPENDIX A

SUPPLEMENTARY MATERIAL TO CHAPTER 4

A.1 Proof of Theorem 4.1

We start by expanding the risk terms in (4.10). The false negative (miss) rate due to early negative decision for the first stage is

$$R_{1,M} = \int \mathrm{p}(\mathrm{d}y_1)\big\{C_M \pi_1(y_1)\mathbb{I}(\delta_1 = 0)\big\} \tag{A.1}$$

where $\mathbb{I}()$ denotes the indicator function that takes value 1 when its argument is true and 0 otherwise. $\mathrm{p}(\mathrm{d}y_{1:K})$ is the probability measure of feature realizations $y_{1:K}$.

Likewise, the miss terms for the stage $i = 2, \ldots, K$ can be given as follows.

$$R_{i,M} = \int \mathrm{p}(\mathrm{d}y_{1:i})\Big\{C_M \pi_i(y_{1:i})\mathbb{I}(\delta_i = 0, \delta_{i-1} = F)\Big\} \tag{A.2}$$

Similarly, the false-alarm (false positive) term at the last stage is given as follows.

$$\begin{aligned} R_{K,A} = \int \mathrm{p}(\mathrm{d}y_{1:K})\Big\{&C_A(1 - \pi_K(y_{1:K})) \\ &\mathbb{I}(\delta_K = 1, \delta_{K-1} = F)\Big\} \end{aligned} \tag{A.3}$$

An important step in solving Problem (4.10) is the following expansion of the expected resource cost in (4.13). By the law of total probability,

$$\begin{aligned} D_1 = D_1\Big\{&\mathrm{P}(\delta_1 = 0) + \sum_{i=2}^{K-1} \mathrm{P}(\delta_i = 0, \delta_{i-1} = F) + \\ &\mathrm{P}(\delta_K = 0, \delta_{K-1} = F) + \mathrm{P}(\delta_K = 1, \delta_{K-1} = F)\Big\} \end{aligned} \tag{A.4}$$

and

$$D_{i+1}\mathrm{P}(\delta_i = F) = D_{i+1}\Big\{ \sum_{j=i+1}^{K-1} \mathrm{P}(\delta_j = 0, \delta_{j-1} = F) + \mathrm{P}(\delta_K = 0, \delta_{K-1} = F) + \mathrm{P}(\delta_K = 1, \delta_{K-1} = F)\Big\}, \quad i = 1, \ldots, K-1 \tag{A.5}$$

Similar expansions can be done for $d_i, i = 2, \ldots, K$, i.e.

$$d_{i+1}\mathrm{P}(\delta_i = 0) = d_{i+1}\Big\{ \sum_{j=2}^{i} \mathrm{P}(\delta_j = 0, \delta_{j-1} = F) + \mathrm{P}(\delta_1 = 0)\Big\}, \quad i = 1, \ldots, K-1 \tag{A.6}$$

Tables A.1 and A.2 illustrate the decomposition of on- and off-resource costs, respectively.

Table A.1: Illustration of the on-resource cost decomposition. An x denotes a valid term.

	$\mathrm{P}(\delta_1 = 0)$	$\mathrm{P}(\delta_2 = 0, \delta_1 = F)$	$\ldots$	$\mathrm{P}(\delta_K = 0, \delta_{K-1} = F)$	$\mathrm{P}(\delta_K = 1, \delta_{K-1} = F)$
D_K				x	x
D_{K-1}				x	x
$\vdots$					
D_2		x		x	x
D_1	x	x		x	x

Table A.2: Illustration of the off-resource cost decomposition. An x denotes a valid term.

	$\mathrm{P}(\delta_1 = 0)$	$\mathrm{P}(\delta_2 = 0, \delta_1 = F)$	$\ldots$	$\mathrm{P}(\delta_{K-1} = 0, \delta_{K-2} = F)$
d_K	x	x		x
d_{K-1}	x	x		
$\vdots$				
d_3	x	x		
d_2	x			

Putting everything back into (4.10) yields a dynamic programming structure, with the state variable being the posteriors π_i defined in Section 4.3.1.

Minimizing (4.10) can thus be achieved efficiently using the following backward procedure.

$$\begin{aligned} V_K(\pi_K) &\triangleq \min_{\delta_K} \mathbb{I}(\delta_K = 0) C_M \pi_K + \mathbb{I}(\delta_K = 1) C_A (1 - \pi_K) \\ V_i(\pi_i) &\triangleq \min_{\delta_i} \mathbb{I}(\delta_i = 0) C_M \pi_i + \\ &\quad \mathbb{I}(\delta_i = F) \left\{ \lambda (D_{i+1} - d_{i+1}) + \mathbb{E}[V_{i+1}(\pi_{i+1}(Y_{i+1}, \pi_i))] \right\} \\ &\quad i = 1, \ldots, K-1 \\ V_0(\pi_0) &\triangleq \lambda \sum_{i=1}^{K} d_i + \lambda (D_1 - d_1) + \mathbb{E}[V_1(\pi_1(Y_1, \pi_0))] \end{aligned} \tag{A.7}$$

where the expectation is taken with respect to the evidence probabilities (see Section 4.3.1) and V_i is the value function at stage i. From the first and second expressions of (A.7), the minimizers for the system can be obtained by setting

$$\delta_K^*(\pi_K) = \begin{cases} 0, \pi_K < C_A/(C_A + C_M) \\ 1, \text{ else} \end{cases} \tag{A.8}$$

and

$$\begin{aligned} \delta_i^*(\pi_i) = \begin{cases} 0, V_i(\pi_i) = C_M \pi_i \\ F, V_i(\pi_i) < C_M \pi_i \end{cases}, \\ i = 1, \ldots, K-1 \end{aligned} \tag{A.9}$$

The expression in (A.9) can be further simplified into (4.14) using Lemmas A.1 and A.3.

Lemma A.1. *$\mathbb{E}[V_{i+1}(\pi_{i+1}(Y_{i+1}, \pi))], i = 0, \ldots, K-1$ and $V_i(\pi), i = 1, \ldots, K$ are concave.*[1]

Proof. $V_K(\pi)$ is concave. Hence, by Lemma A.2, $\mathbb{E}[V_K(\pi_K(Y_K, \pi))]$ is concave.

Assume that $V_{i+1}(\pi)$ is concave, thus $\mathbb{E}[V_{i+1}(\pi_{i+1}(Y_{i+1}, \pi))]$ is concave by Lemma A.2, then

$$V_i(\pi) = \min\{C_M \pi, \lambda (D_{i+1} - d_{i+1}) + \mathbb{E}[V_{i+1}(\pi_{i+1}(Y_{i+1}, \pi))]\} \tag{A.10}$$

[1]Moreover, $V_i(\pi), i = 1, \ldots, K$ can be shown to be piece-wise linear and concave, which was first observed and proven (by induction) in [120, Smallwood and Sondik].

is also concave. Again, by Lemma A.2, $\mathbb{E}[V_i(\pi_i(Y_i, \pi))]$ is concave. □

Lemma A.2. $\mathbb{E}[V_{i+1}(\pi_{i+1}(Y_{i+1}, \pi))]$ *is concave if* $V_{i+1}(\pi)$ *is concave.*

Proof. See [5, p. 146]. □

Lemma A.3. $\mathbb{E}[V_{i+1}(\pi_{i+1}(Y_{i+1}, 0))] = 0, i = 0, \ldots, K-1.$

Proof. $V_K(0) = 0$, then $\mathbb{E}[V_K(\pi_K(Y_K, 0))] = V_K(0) = 0$ and

$$V_{K-1}(0) = \min\{0, \lambda(D_K - d_K)\} = 0 \tag{A.11}$$

Hence, $\mathbb{E}[V_{K-1}(\pi_{K-1}(Y_{K-1}, 0))] = V_{K-1}(0) = 0$

Now assume that $\mathbb{E}[V_{i+1}(\pi_{i+1}(Y_{i+1}, 0))] = 0$, then

$$V_i(0) = \min\{0, \lambda(D_{i+1} - d_{i+1})\} = 0. \tag{A.12}$$

Hence, $\mathbb{E}[V_i(\pi_i(Y_i, 0))] = V_i(0) = 0.$ □

A.2 Proof of Proposition 4.2

Introducing (additional) early positive decisions to intermediate stages results in the following modification to the second expression of (A.7).

$$\begin{aligned} V_i(\pi_i) \triangleq \min_{\delta_i} & \mathbb{I}(\delta_i = 0) C_M \pi_i + \\ & \mathbb{I}(\delta_i = 1) C_A (1 - \pi_i) + \\ & \mathbb{I}(\delta_i = F) \Big\{ \lambda(D_{i+1} - d_{i+1}) + \mathbb{E}[V_{i+1}(\pi_{i+1}(Y_{i+1}, \pi_i))] \Big\} \\ & i = 1, \ldots, K-1 \end{aligned} \tag{A.13}$$

Therefore the positive decision is *not* chosen by the optimal policy under the following circumstances.

$$\begin{aligned} \delta_i^* \neq 1 \text{ if } V_i < C_A(1 - \pi_i), \\ i = 1, \ldots, K-1 \end{aligned} \tag{A.14}$$

Since V_i is a concave function of π_i, (A.14) is equivalent to

$$\begin{aligned} \delta_i^* \neq 1 \text{ if } \pi_i \leq \max\{\pi_i : V_i < C_A(1-\pi_i)\}, \\ i = 1, \ldots, K-1 \end{aligned} \tag{A.15}$$

Hence if (4.21) holds then the positive decisions are never chosen by the optimal policy, and therefore make no difference in the system performance.

A.3 Proof of Proposition 4.3

Recall that the feature model used in the duty-cycling design is the same as that of the cascade's last stage, i.e. the best one. The corresponding miss risk is then given by

$$\begin{aligned} R_{\mathrm{dc},M} &\triangleq \int \mathrm{p}(\mathrm{d}y_K) C_M \pi_K(y_K) \mathbb{I}(\delta_{\mathrm{dc}} = 0) \\ &= \int \mathrm{p}(\mathrm{d}y_{1:K}) C_M \pi_K(y_{1:K}) \mathbb{I}(\delta_{\mathrm{dc}} = 0) \\ &\geq R_{K,M}^* \end{aligned} \tag{A.16}$$

where δ_{dc} is the duty-cycling's detection strategy. The second line follows from the law of total probability and the third one holds by definition. Similarly for the false-alarm risk, i.e.

$$R_{\mathrm{dc},A} \geq R_{K,A}^* \tag{A.17}$$

From (4.22), the duty-cycling system risk is lower-bounded by

$$\rho(R_{K,M}^* + R_{K,A}^* + \lambda D_K) + (1-\rho)(C_M \pi_0 + \lambda d_K) \tag{A.18}$$

assuming zero overhead for duty-cycling, i.e. $D_{\mathrm{dc}} = D_K$ and $d_{\mathrm{dc}} = d_K$. Let ΔR denote the difference between (A.18) and the cascade performance in (4.10). Notice that $\Delta R(\rho)$ is a linear function of ρ. Hence, for the cascade design to outperform the duty-cycling design uniformly ($\Delta R(\rho) \leq 0, \forall \rho$), then $\Delta R(0) \leq 0$ and $\Delta R(1) \leq 0$ must hold.

The inequality $\Delta R(0) \leq 0$ is equivalent to the following trivial condition

on the cascade design

$$R^* \leq C_M \pi_0 + \lambda d_K \tag{A.19}$$

which simply states that the minimal risk achievable by the cascade design must be lower than that of doing nothing (the right-hand side of (A.19)).

On the other hand, the inequality $\Delta R(1) \leq 0$ is equivalent to the following non-trivial condition on the cascade design

$$\lambda E^* + \sum_{i=1}^{K-1} R^*_{i,M} \leq \lambda D_K \tag{A.20}$$

Note that the optimal resource consumption E^* is equal to the resource budget e. Therefore, (A.20) is equivalent to (4.24). The computation of $\sum_{i=1}^{K-1} R^*_{i,M}$ follows directly from Appendix A.1.

A.4 Derivation of the robust transformation on feature/likelihood models

This section archives the derivation of robust likelihood models in Eq. (4.6) based on the uncertainty model introduced by Huber [89],[90, Chapter 10],[91, Chapter 6]. Namely, let $f_1(y), f_0(y)$ be the nominal feature distributions under each hypothesis. Then Huber's uncertainty model is two sets of distributions given as follows.

$$\begin{aligned} \mathcal{G}_0 &\triangleq \{g_0 : G_0(y) \geq (1-\epsilon_0)F_0(y) - \nu_0\} \\ \mathcal{G}_1 &\triangleq \{g_1 : G_1(y) \leq (1-\epsilon_1)F_1(y) + \epsilon_1 + \nu_1\} \end{aligned} \tag{A.21}$$

where uppercase notations are used to denote cumulative versions of the corresponding density/mass probability functions, represented in lowercase, e.g. G_0, G_1 are cumulative versions of a feature distribution pair g_0, g_1, respectively. Huber's minimax approach seeks a robust detector that assumes the worst-case distributions, whose general forms are

$$\begin{aligned} G^*_0(y) &= (1-\epsilon_0)F_0(y) - \nu_0, & l_L \leq l(y) \leq l_U \\ G^*_1(y) &= (1-\epsilon_1)F_1(y) + \epsilon_1 + \nu_1, & l_L \leq l(y) \leq l_U \end{aligned} \tag{A.22}$$

where $l(y) \triangleq f_1(y)/f_0(y)$ is the (nominal) likelihood ratio, and l_L, l_U are its lower and upper bounds, respectively. Outside of $[l_L, l_U]$, it is free to shape the distributions, and constant likelihood ratios are desired, i.e.

$$\begin{aligned} G_1^*(y) &= \frac{1-\epsilon_1}{1-\epsilon_0} l_L G_0^*(y), l(y) < l_L \\ G_1^*(y) &= \frac{1-\epsilon_1}{1-\epsilon_0} l_U G_0^*(y), l(y) > l_U \end{aligned} \tag{A.23}$$

For $l(y) < l_L$, continuity at l_L requires that

$$\begin{aligned} G_0^*(y) &= \alpha[(1-\epsilon_0)F_0(y) - \nu_0] + (1-\alpha)\frac{1-\epsilon_0}{(1-\epsilon_1)l_L}[(1-\epsilon_1)F_1(y) + \epsilon_1 + \nu_1] \\ &= \alpha[(1-\epsilon_0)F_0(y) - \nu_0] + (1-\alpha)\frac{1-\epsilon_0}{l_L}[F_1(y) + \frac{\epsilon_1 + \nu_1}{1-\epsilon_1}] \end{aligned} \tag{A.24}$$

where $\alpha \in [0,1]$ is a free parameter. Letting $\alpha = \frac{v'}{v' + w' l_L}$ (where v', w' are defined in Eq. (4.7)) results in G_0^* being a linear combination of only F_0 and F_1, i.e.

$$G_0^*(y) = \frac{1-\epsilon_0}{v' + w' l_L}[v' F_0(y) + w' F_1(y)], l_i(y) < l_L \tag{A.25}$$

which, after taking the derivative on both sides, is equivalent to (4.6). Finally, a similar result can be obtained for $l(y) > l_U$. This completes the derivation of Eq. (4.6).

A.5 Algorithm implementing the guided-processing principle

The algorithm to optimize a cascade system with the guided-processing principle is given below.

Algorithm 1 Pseudo-code to find optimal thresholds for the cascade system.

function OPTIMIZE(*model*)
    model is a structure containing the system's feature models
    M is the probability quantization size
    $b = [0 : 1/(M-1) : 1]$ (dummy) probability vector
    Use (4.6) to obtain robust versions of *model*.
    $V_K = \min(C_M b, C_A(1-b))$
    $\tau_K^* = C_A/(C_A + C_M)$
    for $i = K-1 : -1 : 1$ **do**
        J = expected next-stage (i+1) value function
        $V_i = \min(C_M b + \lambda \mathbf{d}_{i+1}, J)$
        $\tau_i^* = \min\{b : V_i - (C_M b + \lambda \mathbf{d}_{i+1}) < 0\}$
        $\tau_i^* = \max(\pi_{Li}, \min(\pi_{Ui}, \tau_i^*))$
    end for
    J = expected next-stage (1) value function
    $V_0 = J$
end function

APPENDIX B

SUPPLEMENTARY MATERIAL TO CHAPTER 6

B.1 Proof of Theorem 6.1

We start by expanding the risk terms in (6.4). The false negative (miss) rate due to early negative decision for the first stage is

$$\begin{aligned} R^1_{1,M} &= \int \mathrm{p}(\mathrm{d}y^1_1)\big\{C^1_M \pi^1_1(y^1_1)\mathbb{I}(\delta^1_1 = 0)\big\} \\ R^2_{1,M} &= \int \mathrm{p}(\mathrm{d}y^{1:2}_1)\big\{C^2_M \pi^2_1(y^2_1)\mathbb{I}(\delta^2_1 = 0, \delta^2_0 = F^2) + \\ &\qquad C^2_M \pi^2_1(y^1_1)\mathbb{I}(\delta^2_1 = 0, \delta^2_0 = F^1)\big\} \end{aligned} \tag{B.1}$$

where $R^1_{1,M}, R^2_{1,M}$ are the first-stage miss risk of the primary and secondary applications, respectively. Furthermore, the first term of $R^2_{1,M}$ is due to using the secondary feature y^2_1 ($\delta^2_0 = F^2$), and the second term is due to using the shared (primary) feature y^1_1 ($\delta^2_0 = F^1$). $\mathbb{I}()$ denotes the indicator function that takes value 1 when its argument (a probability event) is true and 0 otherwise. Finally, $\mathrm{p}(\mathrm{d}y_{1:K})$ is the probability measure of feature realizations $y_{1:K}$.

Likewise, the miss terms for the stage $i = 2, \ldots, K$ can be given as follows.

$$\begin{aligned} R^1_{i,M} &= \int \mathrm{p}(\mathrm{d}y^1_{1:i})\Big\{C^1_M \pi^1_i(y^1_{1:i})\mathbb{I}(\delta^1_i = 0, \delta^1_{i-1} = F^1)\Big\} \\ R^2_{i,M} &= \int \mathrm{p}(\mathrm{d}y^{1:2}_{1:i})\Big\{C^2_M \pi^2_i(y^{1:2}_{1:i-1}, y^2_i)\mathbb{I}(\delta^2_i = 0, \delta^2_{i-1} = F^2) + \\ &\qquad C^2_M \pi^2_i(y^{1:2}_{1:i-1}, y^1_i)\mathbb{I}(\delta^2_i = 0, \delta^1_{i-1} = F^1, \delta^2_{i-1} = F^1)\Big\} \end{aligned} \tag{B.2}$$

where the first term of $R^2_{i,M}$ is again due to using the secondary feature y^2_i ($\delta^2_{i-1} = F^2$), and the second term is due to using the shared feature y^1_i ($\delta^2_{i-1} = F^1$ and $\delta^1_{i-1} = F^1$). Similarly, the false-alarm (false positive) term at

the last stage is given as follows.

$$\begin{aligned} R^1_{K,A} = \int \mathrm{p}(\mathrm{d}y^1_{1:K}) \Big\{ & C^1_A(1 - \pi^1_K(y^1_{1:K})) \\ & \mathbb{I}(\delta^1_K = 1, \delta^1_{K-1} = F^1) \Big\} \\ R^2_{K,A} = \int \mathrm{p}(\mathrm{d}y^{1:2}_{1:K}) \Big\{ & C^2_A(1 - \pi^2_K(y^{1:2}_{1:K-1}, y^2_K)) \\ & \mathbb{I}(\delta^2_K = 1, \delta^2_{K-1} = F^2) + \\ & C^2_A(1 - \pi^2_K(y^{1:2}_{1:K-1}, y^1_K)) \\ & \mathbb{I}(\delta^2_K = 1, \delta^1_{K-1} = F^1, \delta^2_{K-1} = F^1) \Big\} \end{aligned} \tag{B.3}$$

An important step in solving Problem (6.4) is the following expansion of the expected resource cost in (6.5). By the law of total probability,

$$\begin{aligned} D^1_1 = D^1_1 \Big\{ & \mathrm{P}(\delta^1_1 = 0) + \sum_{i=2}^{K-1} \mathrm{P}(\delta^1_i = 0, \delta^1_{i-1} = F^1) + \\ & \mathrm{P}(\delta^1_K = 0, \delta^1_{K-1} = F^1) + \mathrm{P}(\delta^1_K = 1, \delta^1_{K-1} = F^1) \Big\} \end{aligned} \tag{B.4}$$

and

$$\begin{aligned} D^1_{i+1} \mathrm{P}(\delta^1_i = F^1) = D^1_{i+1} \Big\{ & \sum_{j=i+1}^{K-1} \mathrm{P}(\delta^1_j = 0, \delta^1_{j-1} = F^1) + \\ & \mathrm{P}(\delta^1_K = 0, \delta^1_{K-1} = F^1) + \mathrm{P}(\delta^1_K = 1, \delta^1_{K-1} = F^1) \Big\}, \\ & i = 1, \ldots, K-1 \end{aligned} \tag{B.5}$$

Similarly for the secondary application

$$\begin{aligned} D^2_{i+1} \mathrm{P}(\delta^2_i = F^2) = D^2_{i+1} \Big\{ & \sum_{j=i+1}^{K-1} \mathrm{P}(\delta^2_j = 0, \delta^2_{j-1} = F^2) + \\ & \mathrm{P}(\delta^2_K = 0, \delta^2_{K-1} = F^2) + \mathrm{P}(\delta^2_K = 1, \delta^2_{K-1} = F^2) \Big\}, \\ & i = 0, \ldots, K-1 \end{aligned} \tag{B.6}$$

Putting everything back into (6.4) yields a dynamic programming structure, with the state variable being the posteriors $\pi^j_i, j = 1, 2$ defined in Section 6.2.1. Minimizing (6.4) can thus be achieved efficiently using the

following backward procedure.

$$
\begin{aligned}
V_K^1(\pi_K^1) &\triangleq \min_{\delta_K^1} \mathbb{I}(\delta_K^1 = 0) C_M^1 \pi_K^1 + \mathbb{I}(\delta_K^1 = 1) C_A^1 (1 - \pi_K^1) \\
V_K^2(\pi_K^2) &\triangleq \min_{\delta_K^2} \mathbb{I}(\delta_K^2 = 0) C_M^2 \pi_K^2 + \mathbb{I}(\delta_K^2 = 1) C_A^2 (1 - \pi_K^2) \\
V_i^1(\pi_i^1) &\triangleq \min_{\delta_i^1} \mathbb{I}(\delta_i^1 = 0) C_M^1 \pi_i^1 + \\
&\qquad \mathbb{I}(\delta_i^1 = F^1) \Big\{ \lambda D_{i+1}^1 + \mathbb{E}[V_{i+1}^1(\pi_{i+1}^1(Y_{i+1}^1, \pi_i^1))] \Big\} \\
V_i^2(\pi_i^2; \pi_i^1) &\triangleq \min_{\delta_i^2} \mathbb{I}(\delta_i^2 = 0) C_M^2 \pi_i^2 + \\
&\qquad \mathbb{I}(\delta_i^{1*} = F^1, \delta_i^2 = F^1) \mathbb{E}[V_{i+1}^2(\pi_{i+1}^2(Y_{i+1}^1, \pi_i^2); \pi_i^1)] + \\
&\qquad \mathbb{I}(\delta_i^2 = F^2) \Big\{ \lambda D_{i+1}^2 + \mathbb{E}[V_{i+1}^2(\pi_{i+1}^2(Y_{i+1}^2, \pi_i^2); \pi_i^1)] \Big\} \\
&\qquad i = 1, \ldots, K-1 \\
V_0^1(\pi_0^1) &\triangleq \lambda D_1^1 + \mathbb{E}[V_1^1(\pi_1^1(Y_1^1, \pi_0^1))] \\
V_0^2(\pi_0^2; \pi_0^1) &\triangleq \min_{\delta_0^2} \mathbb{I}(\delta_0^2 = F^1) \mathbb{E}[V_1^2(\pi_1^2(Y_1^1, \pi_0^2); \pi_0^1)] + \\
&\qquad \mathbb{I}(\delta_0^2 = F^2) \Big\{ \lambda D_1^2 + \mathbb{E}[V_1^2(\pi_1^2(Y_1^2, \pi_0^2); \pi_0^1)] \Big\}
\end{aligned}
\tag{B.7}
$$

where the expectation is taken with respect to the evidence probabilities (see Section 6.2.1) and V_i^j is the value function at stage i of application j. From the first and third expressions of (B.7), the minimizers for the primary application can be obtained by setting

$$
\delta_K^{1*}(\pi_K^1) = \begin{cases} 0, \pi_K^1 < C_A^1/(C_A^1 + C_M^1) \\ 1, \text{ else} \end{cases} \tag{B.8}
$$

and

$$
\begin{gathered}
\delta_i^{1*}(\pi_i^1) = \begin{cases} 0, V_i^1(\pi_i^1) = C_M^1 \pi_i^1 \\ F^1, V_i^1(\pi_i^1) < C_M^1 \pi_i^1 \end{cases}, \\
i = 1, \ldots, K-1
\end{gathered}
\tag{B.9}
$$

The expression in (B.9) can be further simplified into (6.6) using Lemmas A.1 and A.3.

From the second and fourth expressions of (B.7), the optimal decision rule

for the secondary application is

$$\delta_K^{2*}(\pi_K^2) = \begin{cases} 0, \pi_K^2 < C_A^2/(C_A^2 + C_M^2) \\ 1, \text{ else} \end{cases} \tag{B.10}$$

and

$$\begin{aligned} \delta_i^{2*}(\pi_i^2; \pi_i^1) = \begin{cases} 0, V_i^2 = C_M^2 \pi_i^2, \\ F^2, V_i^2 = \lambda D_{i+1}^2 + \mathbb{E}[V_{i+1}^2(Y_{i+1}^2, \pi_i^2; \pi_i^1)] \\ F^1, V_i^2 = \mathbb{E}[V_{i+1}^2(Y_{i+1}^1, \pi_i^2; \pi_i^1)], \pi_i^1 \geq \tau_i^{1*} \end{cases}, \\ i = 0, \ldots, K-1 \end{aligned} \tag{B.11}$$

The expression in (B.11) can be further simplified into (6.7) using Lemma B.1.

Lemma B.1. *If the condition in* (6.8) *holds, then*

$$\begin{aligned} \delta_i^{2*}(\pi_i^2; \pi_i^1) = \begin{cases} 0, V_i^2 = \pi_i^2, \pi_i^1 < \tau_i^{1*} \\ F^2, V_i^2 < \pi_i^2, \pi_i^1 < \tau_i^{1*} \\ 0, V_i^2 = \pi_i^2, \pi_i^1 \geq \tau_i^{1*} \\ F^1, V_i^2 < \pi_i^2, \pi_i^1 \geq \tau_i^{1*} \end{cases}, \\ i = 0, \ldots, K-1 \end{aligned} \tag{B.12}$$

which implies $\delta_i^{2*} \neq F^2$ *when* $\pi_i^1 \geq \tau_i^{1*}$.

Proof. The fourth expression of (B.7) is equivalent to Eq. (B.12) if and only if

$$\mathbb{E}[V_i^2(Y_i^1, \pi_{i-1}^2)] - \mathbb{E}[V_i^2(Y_i^2, \pi_{i-1}^2)] \leq \lambda D_i^2 \tag{B.13}$$

The condition in (B.13) is made satisfied by (6.8) because of Lemma B.2 (note that V_i^2 is concave over π_i^2 for each π_i^1,$i = 1, \ldots, K$). □

Lemma B.2.

$$\begin{aligned} \mathbb{E}[V_i(Y_i^1, \pi_{i-1})] - \mathbb{E}[V_i(Y_i^2, \pi_{i-1})] \leq C_M^2 \left\{ \mathbb{E}[\pi_i(Y_i^1)] - \mathbb{E}[\pi_i(Y_i^2)] \right\} \\ i = 1, \ldots, K \end{aligned} \tag{B.14}$$

Proof. Since $V_i(\pi_i)$ is concave, $V_i'(\pi_i)$ is non-increasing. Furthermore, $V_i'(\epsilon) =$

C_M^2 for some small $\epsilon > 0$. Therefore, $V_i'(\pi_i) \leq C_M^2$, i.e.

$$V_i(\pi_i(Y_i^1)) - V_i(\pi_i(Y_i^2)) \leq C_M^2\big[\pi_i(Y_i^1) - \pi_i(Y_i^2)\big] \tag{B.15}$$

Taking expectation on both size of (B.15) yields (B.14). □

B.2 Proof of Proposition 6.1

Introducing (additional) early positive decisions to intermediate stages results in the following modification to the third and fourth lines of (B.7).

$$\begin{aligned}
V_i^1(\pi_i^1) \triangleq \min_{\delta_i^1} & \mathbb{I}(\delta_i^1 = 0) C_M^1 \pi_i^1 + \mathbb{I}(\delta_i^1 = 1) C_A^1 (1 - \pi_i^1) \\
& \mathbb{I}(\delta_i^1 = F^1)\Big\{\lambda D_{i+1}^1 + \mathbb{E}[V_{i+1}^1(\pi_{i+1}^1(Y_{i+1}^1, \pi_i^1))]\Big\} \\
V_i^2(\pi_i^2; \pi_i^1) \triangleq \min_{\delta_i^2} & \mathbb{I}(\delta_i^2 = 0) C_M^2 \pi_i^2 + \mathbb{I}(\delta_i^2 = 1) C_A^2 (1 - \pi_i^2) \\
& \mathbb{I}(\delta_i^{1*} = F^1, \delta_i^2 = F^1)\mathbb{E}[V_{i+1}^2(\pi_{i+1}^2(Y_{i+1}^1, \pi_i^2); \pi_i^1)] + \\
& \mathbb{I}(\delta_i^2 = F^2)\Big\{\lambda D_{i+1}^2 + \mathbb{E}[V_{i+1}^2(\pi_{i+1}^2(Y_{i+1}^2, \pi_i^2); \pi_i^1)]\Big\} \\
& i = 1, \ldots, K-1
\end{aligned} \tag{B.16}$$

Therefore the positive decision is *not* chosen by the optimal policy under the following circumstances.

$$\begin{aligned}
\delta_i^{j*} \neq 1 \text{ if } V_i^j < C_A^j(1 - \pi_i^j), \\
i = 1, \ldots, K-1, j = 1, 2
\end{aligned} \tag{B.17}$$

Since V_i^1 and V_i^2 are concave functions of π_i^1 and π_i^2, respectively, (B.17) is equivalent to

$$\begin{aligned}
\delta_i^{j*} \neq 1 \text{ if } \pi_i^j \leq \max\{\pi_i^j : V_i^j < C_A^j(1 - \pi_i^j)\}, \\
i = 1, \ldots, K-1, j = 1, 2
\end{aligned} \tag{B.18}$$

Hence if (6.15) holds then the positive decisions are never chosen by the optimal policy, and therefore do not make any difference in the end system performance.

B.3 Algorithm implementing the feature-sharing principle

The algorithm to optimize a multiple application system with feature-sharing is given below.

Algorithm 2 Pseudo-code to find optimal thresholds for the multi-application cascade system. This algorithm has the time complexity of $O(KM^2L)$ and the space complexity of $\max\{O(KM^2), O(M^2L)\}$, where L is the quantization level of the feature models.

function OPTIMIZE($model^1$, $model^2$)
    $model^1$,$model^2$ are structures containing the primary and secondary application's feature models, respectively
    K is the number of stages
    M is the state probability quantization size
    Use (4.6) to obtain robust versions of $model^1$ and $model^2$.
    $b = [0 : 1/(M-1) : 1]$ (dummy) probability vector
    $V_K^1 = \min(C_M^1 b, C_A^1(1-b))$
    $\tau_K^{1*} = C_A^1/(C_A^1 + C_M^1)$
    for $i = 1 : 1 : M$ **do**
        $V_K^2(:, i) = \min(C_M^2 b, C_A^2(1-b))$
        $\tau_K^{2*}(i) = C_A^2/(C_A^2 + C_M^2)$
    end for
    for $i = K-1 : -1 : 1$ **do**
        J^1 = expected next-stage (i+1) primary value function
        $V_i^1 = \min(C_M^1 b, J^1)$
        $\tau_i^{1*} = \min\{b : V_i^1 - C_M^1 b < 0\}$
        $\tau_i^{1*} = \max(\pi_{Li}^1, \min(\pi_{Ui}^1, \tau_i^{1*}))$
        J^{2_1} = expected next-stage (i+1) secondary value function using the shared primary feature
        J^{2_2} = expected next-stage (i+1) secondary value function using the secondary feature
        for $j = 1 : 1 : M$ **do**
            if $b(j) < \tau_i^{1*}$ **then**
                $V_i^2(:, j) = \min(C_M^2 b, J^{2_2}(:, j))$
                $\tau_i^{2*}(j) = \min\{b : V_i^2(:, j) - C_M^2 b < 0\}$
                $\tau_i^{2*}(j) = \max(\pi_{Li}^2, \min(\pi_{Ui}^2, \tau_i^{2*}(j)))$
            else
                $V_i^2(:, j) = \min(C_M^2 b, J^{2_1}(:, j))$
                $\eta_i^{2*}(j) = \min\{b : V_i^2(:, j) - C_M^2 b < 0\}$
                $\eta_i^{2*}(j) = \max(\pi_{Li}^2, \min(\pi_{Ui}^2, \eta_i^{2*}(j)))$
            end if
        end for
    end for
    $V_0^1 = J^1$ = expected next-stage (1) primary value function
    $V_0^2 = J^{2_1}$ = expected next-stage (1) secondary value function using the shared primary feature
end function

APPENDIX C

SUPPLEMENTARY MATERIAL TO CHAPTER 7

C.1 Algorithm for extracting the AI-gram-based TFR

Listing C.1: Extracting AI-gram-based TFR (ridges/regions) in Python

```
1  import numpy as np
2
3  def ridgeTracker(S, noiseFloorInit=2e2, btTime=32e-3,
4      tInc=16e-3):
5
6      btLen = int(btTime/tInc)
7      alp = np.exp(-tInc/btTime)
8
9      nF,nT = np.shape(S)
10     snrCum = np.zeros((nF,nT))
11     fOff = 2 # freq offset
12     supThresh = 2/(1-alp) # suppression threshold
13
14     noiseFloor = np.zeros((nF,nT+1))
15     rUp = 1.01
16     rUpSlow = 1.005
17     rDown = 0.99
18
19     snrAcc = np.zeros(nF)
20     freAcc = np.arange(nF).astype(int)
21     noiseFloor[:,0] = noiseFloorInit*np.ones(nF)
22     ind = btLen*np.ones(nF) # indicator
23     for t in range(nT):
24         snrAccLast = np.array(snrAcc)
```

```
25      for f in range(nF):
26        # noise update
27        if S[f,t] > noiseFloor[f,t]:
28          ind[f] -= 1
29          if ind[f] < 0:
30            noiseFloor[f,t+1] = noiseFloor[freAcc[f],t]*
31              rUp
32          else:
33            noiseFloor[f,t+1] = noiseFloor[freAcc[f],t]*
34              rUpSlow
35        else:
36          noiseFloor[f,t+1] = max(1e-6,
37            noiseFloor[freAcc[f],t]*rDown)
38          ind[f] = btLen
39
40        # snr update
41        snr = max(0.,np.log(S[f,t]**2/noiseFloor[f,t]**2))
42        fLow = max(0,f-fOff)
43        fHigh = min(nF,f+fOff+1)
44        wWin = 1-0.05*np.abs(f-np.arange(fLow,fHigh))/fOff
45        objFun = alp*snrAccLast[fLow:fHigh]+wWin*snr
46        snrAcc[f] = np.max(objFun)
47        freAcc[f] = fLow + np.argmax(objFun)
48
49      # optional max pooling to extract regions
50      # instead of ridges
51       '''
52      snrAccLast = np.array(snrAcc)
53      for f in range(nF):
54        fLow = max(0,f-fOff)
55        fHigh = min(nF,f+fOff+1)
56        fun = snrAccLast[fLow:fHigh]
57        if f == fLow+np.argmax(fun):
58          snrAcc[f] = np.max(fun)
59        else:
60          snrAcc[f] = 0
```

```
61        '''
62
63        # per-bin detection
64        for f in range(nF):
65            if snrAcc[f] > supThresh:
66                snrCum[f,t-btLen//2] = snrAcc[f]-supThresh
67
68    return snrCum
```

REFERENCES

[1] E. A. Lee, J. D. Kubiatowicz, J. M. Rabaey, A. L. Sangiovanni-Vincentelli, S. A. Seshia, J. Wawrzynek, D. Blaauw, P. Dutta, K. Fu, C. Guestrin et al., *The TerraSwarm Research Center (TSRC)(A white paper)*. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2012-207, 2012.

[2] A. A. Chien and V. Karamcheti, "Moore's law: The first ending and a new beginning," *Computer*, no. 12, pp. 48–53, 2013.

[3] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "Soundsense: scalable sound sensing for people-centric applications on mobile phones," in *Proceedings of the 7th International Conference on Mobile systems, Applications, and Services*. ACM, 2009, pp. 165–178.

[4] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, "The Jigsaw continuous sensing engine for mobile phone applications," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2010, pp. 71–84.

[5] D. P. Bertsekas, *Dynamic Programming and Stochastic Control*. Academic Press, Inc., 1976.

[6] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, "Ear-phone: an end-to-end participatory urban noise mapping system," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, 2010, pp. 105–116.

[7] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, "Design of a wireless sensor network platform for detecting rare, random, and ephemeral events," in *Proceedings of the 4th international symposium on Information processing in sensor networks*. IEEE Press, 2005, p. 70.

[8] J. Campbell, P. B. Gibbons, S. Nath, P. Pillai, S. Seshan, and R. Sukthankar, "IrisNet: an internet-scale architecture for multimedia sensors," in *Proceedings of the 13th Annual ACM International Conference on Multimedia*. ACM, 2005, pp. 81–88.

[9] "Array of Things," https://arrayofthings.github.io/index.html, accessed: 2017-03-30.

[10] M. Harris, "A Web of Sensors Enfolds an Entire Forest to Uncover Clues to Climate Change," http://spectrum.ieee.org/green-tech/conservation/a-web-of-sensors-enfolds-an-entire-forest-to-uncover-clues-to-climate-change, 2015, [Online; accessed 2015].

[11] "The Air Quality Egg website," 2014. [Online]. Available: http://www.airqualityegg.com/

[12] "Xively Wikipedia," 2015. [Online]. Available: https://en.wikipedia.org/wiki/Xively

[13] X. Jiang, S. Dawson-Haggerty, P. Dutta, and D. Culler, "Design and implementation of a high-fidelity AC metering network," in *International Conference on Information Processing in Sensor Networks (IPSN), 2009.* IEEE, 2009, pp. 253–264.

[14] C. Mydlarz, J. Salamon, and J. P. Bello, "The implementation of low-cost urban acoustic monitoring devices," *Applied Acoustics, Special Issue on Acoustics of Smart Cities*, vol. 117, pp. 207–218, 2017.

[15] "Siri Wikipedia," 2015. [Online]. Available: https://en.wikipedia.org/wiki/Siri

[16] "Mircosoft Cortana Wikipedia," 2016. [Online]. Available: https://en.wikipedia.org/wiki/Cortana_(software)

[17] "Amazon Echo Wikipedia," 2015. [Online]. Available: https://en.wikipedia.org/wiki/Amazon_Echo

[18] P. Jain, J. Manweiler, and R. R. Choudhury, "Overlay: Practical mobile augmented reality," in *Proceedings of the 13th International Conference on Mobile Systems, Applications, and Services.* ACM, 2015.

[19] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Computer Vision–ECCV 2006.* Springer, 2006, pp. 404–417.

[20] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[21] X. Sheng, J. Tang, X. Xiao, and G. Xue, "Sensing as a service: Challenges, solutions and future directions," *Sensors Journal, IEEE*, vol. 13, no. 10, pp. 3733–3741, 2013.

[22] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by Internet of Things," *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 1, pp. 81–93, 2014.

[23] C. Perera, A. Zaslavsky, C. H. Liu, M. Compton, P. Christen, and D. Georgakopoulos, "Sensor search techniques for sensing as a service architecture for the Internet of Things," *Sensors Journal, IEEE*, vol. 14, no. 2, pp. 406–420, 2014.

[24] J. M. Wolfe, "Guided search 2.0 a revised model of visual search," *Psychonomic Bulletin & Review*, vol. 1, no. 2, pp. 202–238, 1994.

[25] J. M. Wolfe, "Guided search 4.0," *Integrated Models of Cognitive Systems*, pp. 99–119, 2007.

[26] K. Kim, K.-H. Lin, D. B. Walther, M. A. Hasegawa-Johnson, and T. S. Huang, "Automatic detection of auditory salience with optimized linear filters derived from human annotation," *Pattern Recognition Letters*, vol. 38, pp. 78–85, 2014.

[27] "The global data plane," https://swarmlab.eecs.berkeley.edu/projects/4814/global-data-plane, 2015, [Online; accessed 2015].

[28] "The MongoDB 2.6 Manual," http://docs.mongodb.org/manual/, 2014, [Online; accessed 2014].

[29] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 34, no. 4, pp. 744–754, 1986.

[30] D. P. W. Ellis, "Sinewave and sinusoid+noise analysis/synthesis in Matlab," 2003, online web resource. [Online]. Available: http://www.ee.columbia.edu/~dpwe/resources/matlab/sinemodel/

[31] H. K. Kwok and D. L. Jones, "Improved instantaneous frequency estimation using an adaptive short-time Fourier transform," *IEEE Transactions on Signal Processing*, vol. 48, no. 10, pp. 2964–2972, 2000.

[32] C. Dubois and M. Davy, "Joint detection and tracking of time-varying harmonic components: a flexible Bayesian approach," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1283–1295, 2007.

[33] R. R. Tenney and N. R. Sandell Jr, "Detection with distributed sensors," *IEEE Transactions on Aerospace Electronic Systems*, vol. 17, pp. 501–510, 1981.

[34] I. Y. Hoballah and P. K. Varshney, "Distributed Bayesian signal detection," *IEEE Transactions on Information Theory*, vol. 35, no. 5, pp. 995–1000, 1989.

[35] I. Y. Hoballah and P. Varshney, "Neyman-Pearson detection with distributed sensors," in *25th IEEE Conference on Decision and Control, 1986*. IEEE, 1986, pp. 237–241.

[36] M. Gastpar and M. Vetterli, "Source-channel communication in sensor networks," in *Information Processing in Sensor Networks*. Springer Berlin Heidelberg, 2003, pp. 162–177.

[37] M. Gastpar and M. Vetterli, "Power, spatio-temporal bandwidth, and distortion in large sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 745–754, 2005.

[38] J.-F. Chamberland and V. V. Veeravalli, "Decentralized detection in sensor networks," *IEEE Transactions on Signal Processing*, vol. 51, no. 2, pp. 407–416, 2003.

[39] J.-F. Chamberland and V. V. Veeravalli, "Asymptotic results for decentralized detection in power constrained wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 6, pp. 1007–1015, 2004.

[40] A. Dembo and O. Zeitouni, *Large Deviations Techniques and Applications*. Springer Science & Business Media, 2009.

[41] C. Rago, P. Willett, and Y. Bar-Shalom, "Censoring sensors: A low-communication-rate scheme for distributed detection," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 32, no. 2, pp. 554–568, 1996.

[42] S. Joshi and S. Boyd, "Sensor selection via convex optimization," *IEEE Transactions on Signal Processing*, vol. 57, no. 2, pp. 451–462, 2009.

[43] J. Li and G. AlRegib, "Distributed estimation in energy-constrained wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 57, no. 10, pp. 3746–3758, 2009.

[44] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in *Proceedings of the Thirtieth International Conference on Very Large Databases-Volume 30*. VLDB Endowment, 2004, pp. 588–599.

[45] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: an acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 1, pp. 122–173, 2005.

[46] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong, "Approximate data collection in sensor networks using probabilistic models," in *Proceedings of the 22nd International Conference on Data Engineering, 2006. ICDE'06.* IEEE, 2006, pp. 48–48.

[47] O. C. Imer and T. Basar, "Optimal estimation with limited measurements," in *IEEE Conference on Decision and Control and European Control Conference, 2005. CDC-ECC'05. 44th.* IEEE, 2005, pp. 1029–1034.

[48] A. Nayyar, T. Basar, D. Teneketzis, and V. V. Veeravalli, "Optimal strategies for communication and remote estimation with an energy harvesting sensor," *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2246–2260, 2013.

[49] A. O. Hero and D. Cochran, "Sensor management: Past, present, and future," *Sensors Journal, IEEE*, vol. 11, no. 12, pp. 3064–3075, 2011.

[50] D. A. Castanon, "Approximate dynamic programming for sensor management," in *Proceedings of the 36th IEEE Conference on Decision and Control, 1997*, vol. 2. IEEE, 1997, pp. 1202–1207.

[51] J. Evans and V. Krishnamurthy, "Optimal sensor scheduling for hidden Markov model state estimation," *International Journal of Control*, vol. 74, no. 18, pp. 1737–1742, 2001.

[52] V. Krishnamurthy, "Algorithms for optimal scheduling and management of hidden Markov model sensors," *IEEE Transactions on Signal Processing*, vol. 50, no. 6, pp. 1382–1397, 2002.

[53] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of Markov decision processes," *Mathematics of operations research*, vol. 12, no. 3, pp. 441–450, 1987.

[54] J. Wu, Q.-S. Jia, K. H. Johansson, and L. Shi, "Event-based sensor data scheduling: Trade-off between communication rate and estimation quality," *IEEE Transactions on Automatic Control*, vol. 58, no. 4, pp. 1041–1046, 2013.

[55] R. Washburn, M. Schneider, and J. Fox, "Stochastic dynamic programming based approaches to sensor resource management," in *Proceedings of the Fifth International Conference on Information Fusion, 2002*, vol. 1. IEEE, 2002, pp. 608–615.

[56] V. Krishnamurthy and R. J. Evans, "Hidden Markov model multiarm bandits: a methodology for beam scheduling in multitarget tracking," *IEEE Transactions on Signal Processing*, vol. 49, no. 12, pp. 2893–2908, 2001.

[57] J. L. Williams, J. Iii, and A. S. Willsky, "Performance guarantees for information theoretic active inference," in *International Conference on Artificial Intelligence and Statistics*, 2007, pp. 620–627.

[58] S. Appadwedula, V. V. Veeravalli, and D. L. Jones, "Decentralized detection with censoring sensors," *IEEE Transactions on Signal Processing*, vol. 56, no. 4, pp. 1362–1373, 2008.

[59] A. S. Abu-Romeh and D. L. Jones, "Decentralized detection in censoring sensor networks under correlated observations," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, p. 7, 2010.

[60] H. He and P. K. Varshney, "Distributed detection with censoring sensors under dependent observations," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014*. IEEE, 2014, pp. 5055–5059.

[61] R. S. Blum, "Ordering for estimation and optimization in energy efficient sensor networks," *IEEE Transactions on Signal Processing*, vol. 59, no. 6, pp. 2847–2856, 2011.

[62] P. Braca, S. Marano, and V. Matta, "Single-transmission distributed detection via order statistics," *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 2042–2048, 2012.

[63] E. J. Msechu and G. B. Giannakis, "Sensor-centric data reduction for estimation with WSNs via censoring and quantization," *IEEE Transactions on Signal Processing*, vol. 60, no. 1, pp. 400–414, 2012.

[64] Y. Zheng, R. Niu, and P. K. Varshney, "Sequential bayesian estimation with censored data for multi-sensor systems," *IEEE Transactions on Signal Processing*, vol. 62, no. 10, pp. 2626–2641, 2014.

[65] W. P. Tay, J. N. Tsitsiklis, and M. Z. Win, "Asymptotic performance of a censoring sensor network," *IEEE Transactions on Information Theory*, vol. 53, no. 11, pp. 4191–4209, 2007.

[66] A. McAfee, E. Brynjolfsson, T. H. Davenport, D. Patil, and D. Barton, "Big data," *The Management Revolution. Harvard Bus Rev*, vol. 90, no. 10, pp. 61–67, 2012.

[67] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2001*, vol. 1. IEEE, 2001, pp. I–511.

[68] D. S. Turaga, O. Verscheure, U. V. Chaudhari, and L. D. Amini, "Resource management for networked classifiers in distributed stream mining systems," in *Sixth International Conference on Data Mining (ICDM), 2006*. IEEE, 2006, pp. 1102–1107.

[69] L. Le, D. M. Jun, and D. L. Jones, "Energy-efficient detection system in time-varying signal and noise power," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013*. IEEE, 2013, pp. 2736–2740.

[70] Z.-B. Tang, K. R. Pattipati, and D. L. Kleinman, "Optimization of detection networks: Part I - Tandem structures," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, no. 5, pp. 1044–1059, 1991.

[71] P. F. Swaszek, "On the performance of serial networks in distributed detection," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 1, pp. 254–260, 1993.

[72] R. Viswanathan, S. C. Thomopoulos, and R. Tumuluri, "Optimal serial distributed decision fusion," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 24, no. 4, pp. 366–376, 1988.

[73] H. Luo, "Optimization design of cascaded classifiers," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2005*, vol. 1. IEEE, 2005, pp. 480–485.

[74] S. Ravindran, D. V. Anderson, and J. Rehg, "Cascade jump support vector machine classifiers," in *IEEE Workshop on Machine Learning for Signal Processing, 2005*. IEEE, 2005, pp. 135–139.

[75] D. M. Jun and D. L. Jones, "An energy-aware framework for cascaded detection algorithms," in *IEEE Workshop on Signal Processing Systems (SIPS), 2010*. IEEE, 2010, pp. 1–6.

[76] D. M. Jun and D. L. Jones, "Cascading Signal-Model Complexity for Energy-Aware Detection," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 3, no. 1, pp. 65–74, 2013.

[77] Y. Chen, M. Cho, S. Jeong, D. Blaauw, D. Sylvester, and H. S. Kim, "A dual-stage, ultra-low power acoustic event detection system," *IEEE International Workshop on Signal Processing Systems (SiPS)*, 2016.

[78] J. Chen, R. Tan, G. Xing, X. Wang, and X. Fu, "Fidelity-aware utilization control for cyber-physical surveillance systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 9, pp. 1739–1751, 2012.

[79] D. Cohen, "Managing resources on a multi-modal sensing device for energy-aware state estimation," M.S. thesis, University of Illinois at Urbana-Champaign, Electrical and Computer Engineering Department, 2013.

[80] V. C. Raykar, B. Krishnapuram, and S. Yu, "Designing efficient cascaded classifiers: tradeoff between accuracy and cost," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2010, pp. 853–860.

[81] M. Chen, K. Q. Weinberger, O. Chapelle, D. Kedem, and Z. Xu, "Classifier cascade for minimizing feature evaluation cost," in *International Conference on Artificial Intelligence and Statistics*, 2012, pp. 218–226.

[82] E. Ertin, "Polarimetric processing and sequential detection for automatic target recognition systems," Ph.D. dissertation, The Ohio State University, Electrical Engineering Department, 1999.

[83] K. Trapeznikov, V. Saligrama, and D. Castañón, "Multi-stage classifier design," *Machine Learning*, vol. 92, no. 2-3, pp. 479–502, 2013.

[84] K. Trapeznikov and V. Saligrama, "Supervised sequential classification under budget constraints," in *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, 2013, pp. 581–589.

[85] J. Wang, K. Trapeznikov, and V. Saligrama, "An LP for Sequential Learning Under Budgets," in *AISTATS*, 2014, pp. 987–995.

[86] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[87] M. D. Richard and R. P. Lippmann, "Neural network classifiers estimate bayesian a posteriori probabilities," *Neural Computation*, vol. 3, no. 4, pp. 461–483, 1991.

[88] D. M. Jun, L. Le, and D. L. Jones, "Cheap noisy sensors can improve activity monitoring under stringent energy constraints," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP), 2013*. IEEE, 2013, pp. 683–686.

[89] P. J. Huber, "Robust confidence limits," *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, vol. 10, no. 4, pp. 269–278, 1968.

[90] P. J. Huber, "Robust statistics," *International Encyclopedia of Statistical Science*, pp. 1248–1251, 2011.

[91] B. C. Levy, *Principles of Signal Detection and Parameter Estimation*. Springer, 2008.

[92] H. V. Poor et al., "Quickest detection with exponential penalty for delay," *The Annals of Statistics*, vol. 26, no. 6, pp. 2179–2205, 1998.

[93] N. Mor, B. Zhang, J. Kolb, D. S. Chan, N. Goyal, N. Sun, K. Lutz, E. Allman, J. Wawrzynek, E. A. Lee et al., "Toward a global data infrastructure," *IEEE Internet Computing*, vol. 20, no. 3, pp. 54–62, 2016.

[94] L. Ben-Zur, "Developer tool Spotlight - Using Trepn Profiler for Power-Efficient Apps," https://developer.qualcomm.com/blog/developer-tool-spotlight-using-trepn-profiler-power-efficient-apps, 2011, [Online; accessed Oct-2014].

[95] W. J. Leonard, J. Neal, and R. Ratnam, "Variation of Type B song in the endangered Golden-cheeked Warbler (Dendroica chrysoparia)," *The Wilson Journal of Ornithology*, vol. 122, no. 4, pp. 777–780, 2010.

[96] L. Le and D. Jones, "Feature-sharing in cascade detection systems with multiple applications," *IEEE Journal of Selected Topics in Signal Processing*, 2017.

[97] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 1041–1044.

[98] S. DeBruin, B. Ghena, Y.-S. Kuo, and P. Dutta, "Powerblade: A low-profile, true-power, plug-through energy meter," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '15. New York, NY, USA: ACM, 2015. [Online]. Available: http://doi.acm.org/10.1145/2809695.2809716

[99] K. Kaewtip, L. N. Tan, A. Alwan, and C. E. Taylor, "A robust automatic bird phrase classifier using dynamic time-warping with prominent region identification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013*. IEEE, 2013, pp. 768–772.

[100] K. Kaewtip, A. Alwan, C. O'Reilly, and C. E. Taylor, "A robust automatic birdsong phrase classification: A template-based approach," *The Journal of the Acoustical Society of America*, vol. 140, no. 5, pp. 3691–3701, 2016.

[101] W. Chen, M. Hasegawa-Johnson, N. F. Chen, P. Jyothi, and L. R. Varshney, "Clustering-based phonetic projection in mismatched crowdsourcing channels for low-resourced ASR," *WSSANLP 2016*, p. 133, 2016.

[102] J. B. Allen and F. Li, "Speech perception and cochlear signal processing [Life Sciences]," *IEEE Signal Processing Magazine*, vol. 26, no. 4, 2009.

[103] C. Myers, L. Rabiner, and A. Rosenberg, "Performance tradeoffs in dynamic time warping algorithms for isolated word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 6, pp. 623–635, 1980.

[104] L. N. Tan, K. Kaewtip, M. L. Cody, C. E. Taylor, and A. Alwan, "Evaluation of a sparse representation-based classifier for bird phrase classification under limited data conditions." in *Interspeech*, 2012, pp. 2522–2525.

[105] K. Kaewtip, L. N. Tan, C. E. Taylor, and A. Alwan, "Bird-phrase segmentation and verification: A noise-robust template-based approach," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015*. IEEE, 2015, pp. 758–762.

[106] L. N. Tan, A. Alwan, G. Kossan, M. L. Cody, and C. E. Taylor, "Dynamic time warping and sparse representation classification for birdsong phrase classification using limited training data," *The Journal of the Acoustical Society of America*, vol. 137, no. 3, pp. 1069–1080, 2015.

[107] M. S. Lewicki, "Efficient coding of natural sounds," *Nature Neuroscience*, vol. 5, no. 4, pp. 356–363, 2002.

[108] W. Zeng and R. Church, "Finding shortest paths on real road networks: the case for A*," *International Journal of Geographical Information Science*, vol. 23, no. 4, pp. 531–543, 2009.

[109] I. Pohl, *First Results on the Effect of Error in Heuristic Search*. Edinburgh University, Department of Machine Intelligence and Perception, 1969.

[110] J. Pearl, *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Pub. Co., Inc., Reading, MA, 1984.

[111] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[112] J. W. Schnupp and A. J. King, "Neural processing: the logic of multiplication in single neurons," *Current Biology*, vol. 11, no. 16, pp. R640–R642, 2001.

[113] J. L. Peña and M. Konishi, "Auditory spatial receptive fields created by multiplication," *Science*, vol. 292, no. 5515, pp. 249–252, 2001.

[114] D. Gabor, "Theory of communication. part 1: The analysis of information," *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering*, vol. 93, no. 26, pp. 429–441, 1946.

[115] J. G. Arriaga, M. L. Cody, E. E. Vallejo, and C. E. Taylor, "Bird-db: A database for annotated bird song sequences," *Ecological Informatics*, vol. 27, pp. 21–25, 2015.

[116] M. Likhachev, G. J. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in *NIPS*, 2003, pp. 767–774.

[117] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[118] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[119] H. R. Sheikh, "Image quality assessment using natural scene statistics," Ph.D. dissertation, The University of Texas at Austin, Electrical and Computer Engineering, 2004.

[120] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable Markov processes over a finite horizon," *Operations Research*, vol. 21, no. 5, pp. 1071–1088, 1973.